UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

DISSENY D'UN SISTEMA DE VISIÓ SUBMARINA AMB COMUNICACIÓ ACÚSTICA

Memòria i Annexos

Autor: Oriol Solé Font
Director: Spartacus Gomariz Castro
Co-Director: Ivan Masmitjà Rusiñol

Convocatòria: Maig 2019

Resum

En aquest projecte s'implementa un sistema de visió submarina amb comunicació acústica, amb la finalitat de capturar i transmetre imatges mitjançant un microcontrolador del tipo Raspberry i mòdems acústics, així com estudiar el seu consum energètic i la seva viabilitat.

La comunicació entre la Raspberry i el mòdem acústic transmissor encarregat de l'enviament de les imatges, així com entre el mòdem acústic receptor i PC, destinatari de les imatges, es du a terme a través del protocol de comunicacions TCP/IP. La comunicació entre els mòdems es fa a través del protocol S2C propietari d'EvoLogics.

El governament de la Raspberry, la càmera, així com el tractament de les imatges i la configuració dels paràmetres del canal de comunicació entre el microcontrolador/PC i els mòdems acústics es realitza mitjançant el llenguatge de programació Python.

Per tal de demostrar el correcte funcionament del sistema de visió i comprovar l'assoliment dels objectius inicials del projecte, es realitzen proves tant amb mòdems virtuals, a través d'un servidor ofert per EvoLogics, així com proves amb els mòdems físics, permetent així la contrastació i anàlisi de les dades.

Finalment, cal destacar que els apartats han estat tots resolts satisfactòriament i el desenvolupament d'aquest projecte podrà permetre ser utilitzat en diferents àrees de treball i per diferents aplicacions, com ara dotar al submarí autònom GUANAY II de la UPC d'un sistema de visió submarina, així com permetre a l'Institut de Ciències del Mar de Barcelona tenir un major control dels seus estudis.

Resumen

En este proyecto se implementará un sistema de visión submarina con comunicación acústica, con la finalidad de capturar y transmitir imágenes mediante un microcontrolador del tipo Raspberry y módems acústicos, así como estudiar su consumo energético y su viabilidad.

La comunicación entre la Raspberry y el módem acústico transmisor encargado del envio de las imágenes, así como entre el módem acústico receptor y el PC, destinatario de las imágenes se realiza a través del protocolo de comunicaciones TCP/IP. La comunicación entre los módems se realiza a través del protocolo S2C propietario de EvoLogics.

El control de la Raspberry, la cámara, así como el tratamiento de las imágenes y la configuración de los parámetros del canal de comunicación entre el microcontrolador/PC y los módems acústicos se realiza mediante el lenguaje de programación Python.

Para demostrar el correcto funcionamiento del sistema de visión y comprobar el logro de los objetivos iniciales del proyecto, se realizan pruebas tanto con módems virtuales, mediante el servidor ofrecido por EvoLogics, así como pruebas con los módems físicos, permitiendo así la contrastación y análisis de los datos.

Finalmente, cabe destacar que los apartados han sido todos completamente resueltos satisfactoriamente y el desarrollo de este proyecto podrá permitir ser utilizado en diferentes áreas de trabajo y para distintas aplicaciones, como por ejemplo dotar al submarino autónomo GUANAY II de la UPC de un sistema de visión submarina, así como permitir al Instituto de Ciencias del Mar de Barcelona tener un mayor control en sus estudios.

Abstract

In this Project a submarine vision system with acoustic communication is implemented, with the aim of capturing and transmitting images using a microcontroller of the Raspberry type and acoustic modems, as well as studying its energy consumption and its viability.

The communication between the Raspberry and the acoustic transmitter modem responsible for the sending of the images, as well as between the acoustic modem receiver and PC, the recipient of the images, is carried out through the TCP/IP communications protocol. The communication between both modems is carried out via the S2C protocol by EvoLogics.

The management of Raspberry, the camera, as well as the treatment of the images and the configuration of the parameters of the communication channel between the microcontroller/PC and the acoustic modems is done with the Python programming language.

In order to demonstrate the correct operation of the vision system and to verify the achievement of the initial objectives of the project, tests are carried out both with virtual modems, through a server offered by EvoLogics, the manufacturer of the modems, as well as tests with physical modems, thus allowing the contrasting and analysis of the data and the results.

Finally, it should be noted that the sections have all been satisfactorily resolved and the development of this project will be able to be used in different working areas and for different applications, such as providing the autonomous GUANAY II submarine of the UPC with a vision system submarine, as well as allowing the Institute of Sciences of the Sea of Barcelona to have a greater control of their studies.

Agraïments

En primer lloc, el meu sincer agraïment pel meu tutor Spartacus Gomariz. Gràcies a la seva ajuda, suport i implicació absoluta, aquest projecte s'ha pogut dur a terme exitosament.

A continuació, m'agradaria agrair també al doctorand Ivan Masmitjà, ja que els seus consells, comentaris i suport durant tot el projecte, m'han permès avançar i dotar-me d'altres perspectives que han permès millorar en gran mesura el projecte.

Finalment, donar les gràcies a la meva família, als meus amics i a la meva parella, per animar-me i donar-me el seu suport quan més ho necessitava.

Glossari

ADC: l'acrònim ADC fa referència a les sigles en anglès "Analog-to-Digital Converter", és a dir, un convertidor analògic-digital el qual té la capacitat de convertir un senyal analògic en un altre senyal digital.

DAC: "Digital-to-Analog Converter", es tracta d'un dispositiu electrònic capaç de convertir un senyal digital (generalment binari) en un senyal analògic.

Senyal portadora: és un senyal generalment sinusoïdal, el qual disposa d'algun dels seus paràmetres (amplitud, fase o freqüència) modificat per un altre senyal anomenat moduladora amb l'objectiu de transmetre una informació.

Senyal moduladora: és un senyal generalment binari, el qual disposa de la informació que es vol transmetre i s'utilitza per modificar algun paràmetre de la senyal portadora per així transmetre la informació amb les mínimes pèrdues.

FPGA: "field-programmable gate array" és una matriu de portes programables que a partir d'un llenguatge de descripció d'alt nivell, es poden recrear tant funcions senzilles com les que realitza una porta lògica fins a circuits altament complexos.

USB DAQ: és un dispositiu electrònic l'objectiu del qual és l'adquisició de dades per tal de mesurar senyals elèctrics i convertir aquests en senyals digitals per posteriorment ser presentades en un PC.

QAM: modulació d'amplitud en quadratura, és una tècnica de modulació la qual es basa en transportar dos senyals independents mitjançant la modulació de la senyal portadora tant en amplitud com en fase.

PSK: és una de les tècniques de modulació més utilitzades, basada en variar la fase de la portadora entre un número determinat de valors discrets.

OFDM: multiplexació per divisió de freqüències ortogonals, es tracta d'una tècnica de transmissió d'informació que consisteix en la multiplexació d'un conjunt d'ones portadores de diferents freqüències, on cada una d'elles transporta informació i són modulades en PSK o QAM.

DPSK: és una tècnica de modulació per desplaçament diferencial de fase, on la informació binaria de l'entrada està composada per la diferència entre les fases de dos elements successius de senyalització.

Índex

1. Prefaci

1.1. Origen del treball

Aquest projecte sorgeix de la necessitat de satisfer dos objectius clarament establerts, per una banda, dotar al submarí del grup de recerca SARTI de la UPC, el GUANAY II d'un sistema de visió submarina per tal de capturar i transmetre imatges i així tenir un major control del mateix. Per altre banda, col·laborar amb l'Institut de Ciències del Mar de Barcelona, amb projectes els quals treballen amb gàbies d'escamarlans, per així poder controlar l'obertura i el tancament d'aquestes a través d'imatges, permetent tenir un major control de la situació.

1.2. Motivació

En qualsevol aplicació industrial les comunicacions juguen un paper imprescindible en el correcte desenvolupament dels projectes. Transmetre informació a través de senyals ha estat i és un gran avanç tecnològic. Aquesta tecnologia permet enviar ordres a dispositius, transmetre dades, intercanviar informació, obtenir un major control de certes situacions així com infinitat de possibilitats.

Tot i així, la transmissió de dades a través de senyals és molt diferent segons el medi pel qual es destini l'aplicació en concret. En el medi aquàtic les radiacions electromagnètiques no són eficaces degut a l'atenuació dels senyals i existeixen altres mètodes alternatius per transmetre dades. Un dels mètodes més efectius és a través de senyals acústics a partir dels quals es pot transmetre informació com, per exemple, imatges.

Així doncs, la realització d'aquest projecte ha estat marcada pel gran interès d'aprenentatge dins de l'àrea de comunicacions i més específicament per la possibilitat de treballar amb comunicacions submarines mitjançant senyals acústiques.

1.3. Requeriments previs

Prèviament a la realització del projecte és necessari disposar de coneixements sobre electrònica, ja que serà necessari seleccionar components electrònics en funció de les seves característiques tècniques, així com alimentar-los. També és necessari dominar llenguatges de programació d'alt nivell per a microcontroladors, en concret Python, ja que la major part del treball recau en la programació d'un microcontrolador del tipus Raspberry. Finalment, cal tenir coneixements bàsics sobre el protocol de comunicacions TCP/IP.

2. Introducció

En aquest projecte s'implementarà un sistema de visió submarina amb comunicació acústica que permetrà capturar imatges a través d'una càmera governada per un microcontrolador i transmetre aquestes imatges mitjançant mòdems acústics.

Per dur a terme aquest sistema, primerament es realitzarà un disseny previ i posteriorment es seleccionaran els components en funció de les necessitats i els objectius principals a satisfer. Posteriorment, es realitzarà una programació dels protocols de comunicació entre la càmera i el mòdem ajustant els seus paràmetres al volum d'informació i la capacitat de transmissió del canal.

També per tal de poder estudiar la viabilitat del sistema es realitzarà un estudi dels consums energètics del mateix, per tal de estimar la vida útil i es mostrarà les gràfiques associades a aquests consums.

2.1. Objectius del treball

2.1.1. Objectiu General

Dissenyar un sistema de visió submarina amb comunicació acústica, a partir d'un microcontrolador i mòdems acústics.

2.1.2. Objectius específics

El projecte es basa principalment en quatre parts fonamentals. Una primera part basada en la recopilació d'informació per decidir quin microcontrolador és l'adient pel projecte, així com quina càmera, el tipus de comunicació utilitzada i els components necessaris. Una segona part destinada a la programació, basada en la captura, el tractament i el processament d'imatges a través d'un llenguatge de programació d'alt nivell. Una tercera part de comunicacions, destinada a la comunicació entre els mòdems acústics i el microcontrolador i PC, així com la comunicació acústica per l'enviament de les imatges entre els mòdems. Finalment, una quarta i última part destinada a l'estudi dels consums energètics, ja que per aquest projecte és vital conèixer els consums per controlar el temps que estarà operatiu el sistema.

2.2. Abast del treball

El treball ha de permetre dissenyar un sistema capaç de transmetre imatges mitjançant senyals acústics que permeti millorar el control associat a tasques marines, amb un consum energètic reduït per tal d'optimitzar l'autonomia del sistema.

3. Marc teòric

Aquest capítol té l'objectiu d'introduir i donar context al projecte, a partir de conceptes i definicions que permetran tant facilitar i assegurar la correcte comprensió de qualsevol lector no relacionat amb aquest camp en particular, com també establir unes bases teòriques en el projecte, referent a comunicacions acústiques submarines, i més específicament als mòdems acústics.

3.1. Comunicació acústica submarina

Actualment, les comunicacions no fixes a curta i mitja distància en medis submarins es fan predominantment amb senyals acústics. Una alternativa serien els cables submarins per transmetre informació mitjançant senyals elèctrics però no són una opció econòmica ni tampoc pràctica, especialment quan es tracta de plataformes mòbils. Les ones electromagnètiques tampoc són una bona alternativa ja que en el medi aquàtic es veuen altament atenuades degut a què l'aigua és un medi conductor i provoca grans alteracions en les radiacions electromagnètiques. La utilització de senyals òptics, que en el medi terrestre són una bona opció per transmetre informació i dades, no són gaire útils en el medi aquàtic ja que en distàncies superiors als 200 m les freqüències òptiques es veuen fortament afectades per l'efecte de la dispersió, provocant que el medi es torni pràcticament opac, dificultant i limitant molt la utilització de les mateixes.

La comunicació acústica submarina és una tècnica molt utilitzada avui dia per enviar i rebre informació sota l'aigua a través d'ones acústiques. Aquesta tècnica utilitza generalment els mateixos mètodes de modulació (mètodes per transportar informació sobre una ona portadora, en aquest cas concret, acústica) que els utilitzats en les comunicacions de radio, com per exemple:

- **Modulació per desplaçament de freqüència (FSK):** per a cada nivell binari (0-1 de la senyal digital moduladora) es modula la senyal portadora en dos freqüències concretes.
- **Modulació per desplaçament de fase (PSK):** es tracta d'una forma de modulació angular que consisteix bàsicament en variar la fase de la portadora entre un nombre determinat de valors discrets.
- **Modulació per salt d'espectre expandit (FHSS):** consisteix en emetre una senyal sobre una sèrie de freqüències aleatòries, saltant de forma sincrònica de freqüència en freqüència.
- **Modulació per espectre de propagació de seqüencia directa (DSSS):** s'utilitza per tal de modular una portadora de manera que s'augmenti l'ample de banda i es redueixi la densitat de potència espectral (s'utilitza principalment per reduir el soroll en la comunicació).

- **Multiplexatge per divisió ortogonal de freqüència (OFDM):** consisteix en enviar un conjunt d'ones portadores de diferents freqüències, on cada ona portadora aporta i transporta informació diferent, modulades generalment per desplaçament de fase.
- **Modulació per espectre expandit amb propagació d'escombrat (S2C):** consisteix en una modulació basada en la tècnica FHSS, però amb la implementació d'una portadora que consisteix en una successió d'escombrats que provoquen una ràpida fluctuació permanent de la freqüència del senyal. Aquesta tècnica de modulació permet incrementar la velocitat i l'eficiència de la transmissió d'informació.

Tot i així, la comunicació acústica també presenta una sèrie de dificultats en el medi aquàtic com ara:

- **Propagació de múltiples trajectòries:** en el medi aquàtic existeix el fenomen dels rebots, en que un senyal pot seguir diferents camins i trajectòries en la seva propagació (degut a rebots tant en la superfície com en el fons marí) provocant pèrdues i dificultats en la transmissió de dades.

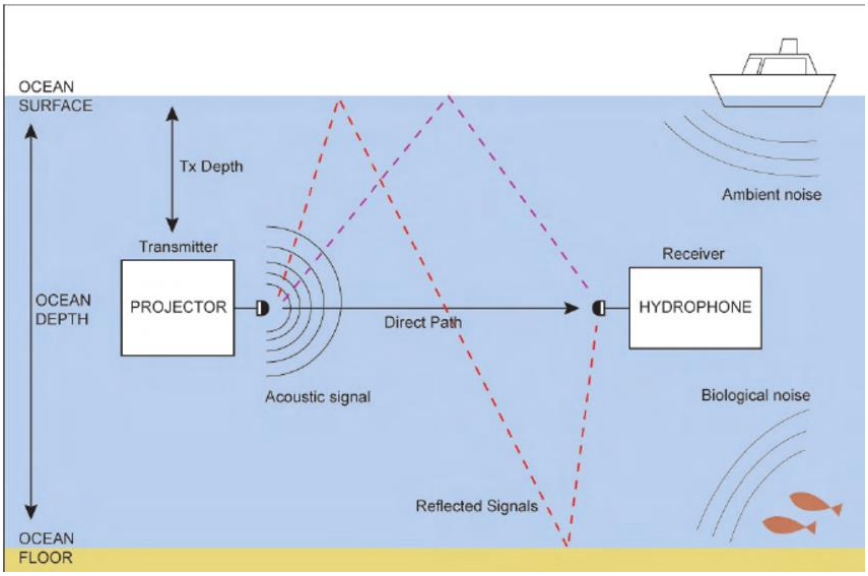

Figura 3.1. Propagació de senyal acústic en múltiples trajectòries. [1]

- **Variacions de temps del canal:** quan s'estableixen canals de comunicacions acústics, el medi aquàtic influeix en el temps de transmissió de dades degut principalment al soroll present, tant ambiental com originat.
- **Alta atenuació dels senyals:** el medi subaquàtic presenta unes propietats que atenuen fortament qualsevol senyal, disminuint així la seva potència. Tot i així, encara que els senyals acústics experimentin aquesta atenuació, ho fan en un menor nivell que els senyals electromagnètics. Com exemple concret, en una freqüència de 30 kHz, l'atenuació que pateix

un senyal electromagnètic és del ordre de $db \cdot m^{-1}$ en canvi un senyal acústic la pateix del ordre de $db \cdot km^{-1}$.

- **Petit ample de banda disponible**
- **Capes de bombolles d'oxigen:** la interacció de les ones acústiques amb aquestes capes de bombolles d'oxigen provoquen l'efecte de dispersió de la ona i per tant ocasiona que aquestes pateixin desviacions de trajectòries.
- **Refraccions per canvis de temperatura i pressió:** les ones acústiques poden patir canvis en les seves trajectòries degut a variacions de la temperatura i la pressió de l'aigua.
- **Efecte Doppler:** la freqüència de les ones acústiques poden variar pel moviment relatiu de l'emissor respecte al receptor, provocant pèrdues de dades i informació.

3.2. Mòdems acústics

3.2.1. Principi de funcionament

Els mòdems acústics són dispositius electrònics utilitzats per transmetre dades sota l'aigua mitjançant senyals acústics. El principi de funcionament és semblant al que utilitzen els mòdems telefònics per transmetre dades a través de línies telefòniques. A partir de dades digitals el mòdem acústic transmissor converteix aquestes dades en senyals sonors especials que són rebudes per un segon mòdem acústic receptor que els converteix un altre cop en dades digitals.

El procés per transmetre un senyal acústic amb la informació d'un senyal digital és fa a través de la modulació, anteriorment explicada. Els mòdems acústics generalment utilitzen una modulació de fase en el transmissor i una desmodulació de fase en el receptor, utilitzant un senyal portador per optimitzar la quantitat d'informació enviada i per minimitzar els efectes del soroll i interferències del medi aquàtic. No obstant això, hi ha altres mètodes com la modulació per espectre expandit, també força utilitzat.

Figura 3.2. Exemple de mòdem acústic subaquàtic comercial. [2]

Tot i que els mòdems acústics presenten les dificultats mencionades en l'apartat anterior (veure apartat 3.1 Comunicació acústica submarina), pel que fa a la transmissió de senyals acústics en un medi subaquàtic, són els dispositius més òptims i utilitzats per transmetre dades sota l'aigua. Tanmateix, els mòdems acústics actuals tan sols poden transmetre dades punt a punt, amb una baixa-mitja velocitat de transmissió de dades i amb una tolerància mitjana al retard.

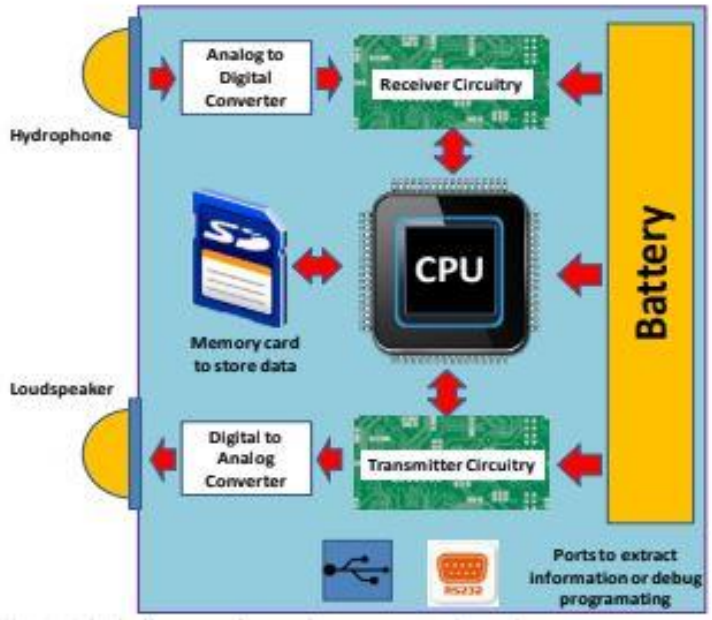

Figura 3.3. Diagrama de blocs d'un mòdem acústic subaquàtic. [4]

En la figura 3.3 es mostra el diagrama de blocs d'un mòdem acústic submarí, representant els seus components interns més característics. Primerament, qualsevol mòdem acústic té un hidròfon que actua com a transductor de so a electricitat, permetent obtenir així qualsevol ona acústica (emesa per un altre mòdem) i transformant la ona a un senyal analògic. A continuació, mitjançant un ADC es converteix el senyal elèctric analògic en un senyal digital, per a ser posteriorment rebut per la unitat central de processament (CPU).

La CPU està composada per un microprocessador i una memòria interna, en què aquesta generalment en els mòdems acústics va acompanyada d'una FPGA i a més solen disposar també de memòries externes. A través d'un DAC es converteix el senyal digital rebut a través de la CPU en un senyal analògic per posteriorment ser transformat en un senyal acústic a través d'un transductor electroacústic, convertint els senyals elèctrics en so.

Els mòdems acústics subaquàtics disposen també d'una unitat de potència amb convertidors DC/DC per ajustar l'alimentació. Per tal de poder extreure/introduir dades o configurar els mòdems, aquests disposen generalment de ports USB, RS-232 i Ethernet.

Aquests dispositius poden ser utilitzats per diverses aplicacions com ara telemetria subaquàtica, per mesurar certes magnituds físiques com ara la pressió o la temperatura de l'aigua, comunicacions, monitoritzacions submarines i registres de dades o inclús comandament i control de vehicles operats a distància (ROV) o vehicles autònoms submarins (AUV) a través dels senyals acústics que emeten.

3.2.2. Models comercials

Actualment en el mercat existeixen diverses marques que comercialitzen aquests tipus de dispositius, les quals presenten diversos models en funció de les possibles necessitats dels clients. En la taula 3.1 es mostra una comparativa de diferents paràmetres com són el tipus de modulació de la informació, la freqüència de la portadora, la velocitat de transmissió de les dades, els consums energètics en la transmissió i recepció, així com la distància màxima de funcionament, entre diversos models comercials de diferents marques.

Taula 3.1. Models comercials de mòdems acústics subaquàtics.

Model	Modulació	Freqüència portadora (kHz)	Velocitat de dades (bps)	Consum energètic transmissió (W)	Consum energètic recepció (W)	Màxima distància de funcionament (m)
Aquatec AQUAModem 1000	FSK DPSK	9,75	2000	20	0,6	5000
DSPComm AquaComm Marlin	FSK DPSK	23	480	1,8	0,252	1000
Evologics S2CM 18/34	S2C	18-34	13900	35	0,8	3500
Evologics S2CR 48/78	S2C	48-78	31200	18	1,1	1000
LinkQuest UWM1000	FSK	35695	17800	1	0,75	3500
LinkQuest UWM4000	FSK	17	8500	7	0,8	4000
Teledyne Marine 910 Series ATM-916	PSK	22-27	15360	20	0,768	6000
AquaSent AM-D2000	OFDM	9-15	375-1500	25	0,8	5000

4. Disseny del sistema de visió submarina

Per tal de dissenyar el sistema de visió submarina, primerament es presenta un esquema dels requisits i components mínims per implementar la funcionalitat del sistema i complir amb els objectius establerts. Posteriorment, es seleccionen els components i es presenten les raons pels quals han estat seleccionats.

4.1. Esquema previ

En aquest projecte, tot i tenir els mòdems acústics ja establerts i definits, ja que són els únics dels quals es disposava, cal ser coherent amb la selecció de la resta de components i intentar seleccionar els que millor s'adaptin tant amb els requisits marcats a l'inici del projecte com amb l'objectiu i finalitat del mateix.

El disseny del sistema ha d'incloure un microcontrolador el qual mitjançant una càmera posteriorment seleccionada, es capturaran imatges i s'enviaran al mòdem acústic transmissor mitjançant el protocol de comunicacions TCP/IP sobre Ethernet, ja que és el protocol que utilitzen els mòdems disponibles. Mitjançant els senyals acústics emesos pel mòdem transmissor s'enviaran les imatges al mòdem acústic receptor, el qual finalment i també a través d'Ethernet, enviarà les imatges a un ordinador.

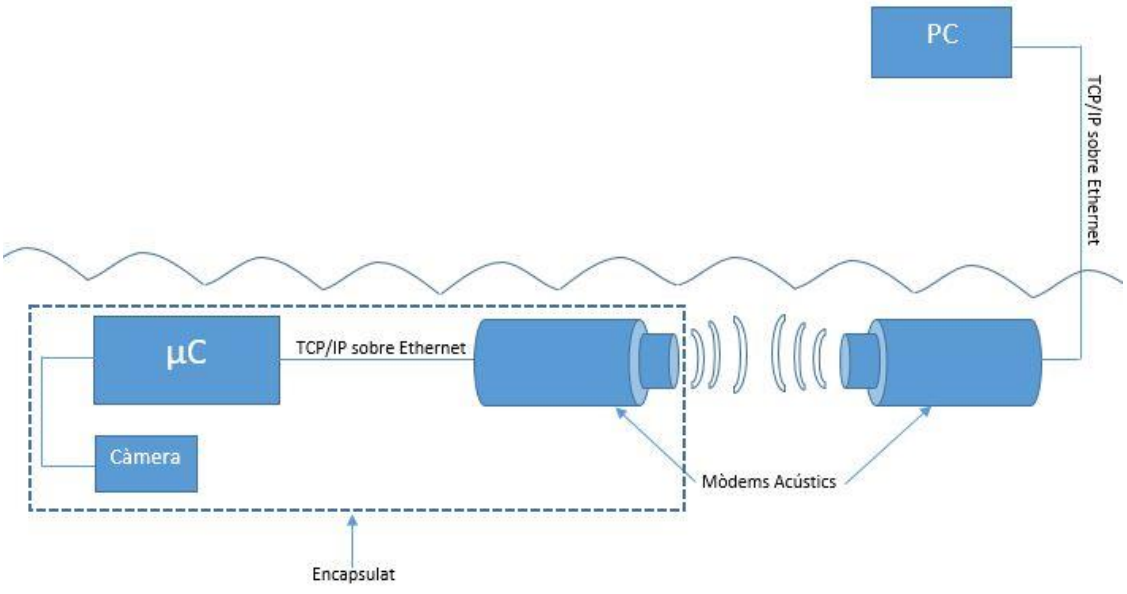

Figura 4.1. Esquema del disseny previ del connexionat i comunicacions del sistema de visió submarina (sense alimentacions).

4.2. Selecció dels components

A partir de l'esquema previ del disseny del sistema de visió submarina i amb els objectius marcats a l'inici del projecte i tenint en compte les limitacions imposades pels mòdems, a continuació es realitza una selecció dels components que formaran part del sistema dissenyat.

4.2.1. Microcontrolador/mini ordinador

El microcontrolador/mini ordinador seleccionat pel projecte és una Raspberry Pi. Aquesta elecció ha estat basada principalment en què aquesta computadora de placa reduïda de baix cost és altament accessible, disposa de programari de codi obert, un sistema operatiu oficial anomenat Raspbian, el qual està basat en la variant Debian del sistema operatiu GNU/Linux, i existeix una gran documentació referent a aquest dispositiu.

A més, el seu sistema operatiu està encara en desenvolupament de forma activa, fet que implica una constant millora de les seves prestacions i característiques. Aquesta computadora de placa reduïda incorpora a més una tecnologia denominada "PIXEL" ("Pi Improved X-Window Environment, Lightweight") com el seu entorn d'escriptori principal, el qual utilitza una interfície gràfica molt senzilla que intenta facilitar el seu ús a l'usuari, a diferència d'altres computadors de placa reduïda els quals són més complexos d'utilitzar.

Certs models de la Raspberry Pi també incorporen un port Ethernet, necessari per la comunicació amb el mòdem acústic i un connector adaptat, pensat per la connexió d'un mòdul de càmera de la pròpia marca, el qual facilita en gran part la connexió i utilització de la mateixa.

Entre tots els models existents de la marca Raspberry Pi, a nivell de hardware, hi ha una sèrie de components que estan inclosos en tots ells: un processador Broadcom, ports USB, HDMI, pins GPIO, així com un connector per una càmera. En la figura 4.2 es pot observar els components genèrics enumerats, així com la seva distribució en la Raspberry Pi 2 model B, els quals corresponen als components principals de tots els models:

- **1:** Ports UBS
- **2:** Pins GPIO
- **3:** CPU/GPU Broadcam
- **4:** Port HDMI
- **5:** Connector per càmera

Figura 4.2. Distribució dels components de la placa 2 model B.

Tot i que els models comparteixen certes especificacions, per tal de seleccionar el model que s'ajusti millor a les necessitats del projecte, s'han tenir en compte les diferents característiques dels models existents de la Raspberry Pi:

- **Característiques de Hardware**: Una primera diferenciació entre els diversos models disponibles de Raspberry Pi és el hardware. A continuació, tal i com es pot observar en la figura 4.3, es mostra una comparació entre els diferents models a nivell d'especificacions de hardware.

Product	SoC	Speed	RAM	USB Ports	Ethernet	Wireless	Bluetooth
Raspberry Pi Model A+	BCM2835	700MHz	512MB	1	No	No	No
Raspberry Pi Model B+	BCM2835	700MHz	512MB	4	100Base-T	No	No
Raspberry Pi 2 Model B	BCM2836/7	900MHz	1GB	4	100Base-T	No	No
Raspberry Pi 3 Model B	BCM2837A0/B0	1200MHz	1GB	4	100Base-T	802.11n	4.1
Raspberry Pi 3 Model B+	BCM2837B0	1400MHz	1GB	4	1000Base-T	802.11ac/n	4.2
Raspberry Pi Zero	BCM2835	1000MHz	512MB	1	No	No	No
Raspberry Pi Zero W	BCM2835	1000MHz	512MB	1	No	802.11n	4.1
Raspberry Pi Zero WH	BCM2835	1000MHz	512MB	1	No	802.11n	4.1

Figura 4.3. Comparació de Hardware dels diferents models de Raspberry Pi. [5]

En la figura 4.3 es pot observar que només els models B+, 2 model B, 3 model B i 3 model B+ disposen d'un port de comunicació Ethernet, requisit indispensable per la realització del projecte, ja que la comunicació entre la Raspberry Pi i el mòdem acústic es realitzarà a través d'Ethernet, per tant, els altres models queden descartats. Altres paràmetres com ara el Bluetooth, la connexió Wireless o el número de ports USB són indiferents, ja que no seran utilitzats, així doncs no es tenen en compte per la selecció del model.

Per altre banda, la RAM i la velocitat de la CPU, si són uns paràmetres força importants pel projecte, i per tant es descarta el model B+ per insuficiència de RAM.

- **Consum energètic:** En tractar-se d'un sistema alimentat per bateria, un altre factor molt important a tenir en compte en la selecció del model de Raspberry Pi és el consum energètic que tenen, ja que per tal de què aquest sistema pugui aguantar el màxim temps possible en funcionament és imprescindible que el model sigui de baix consum.

Product	Recommended PSU current capacity	Maximum total USB peripheral current draw	Typical bare-board active current consumption
Raspberry Pi Model A	700mA	500mA	200mA
Raspberry Pi Model B	1.2A	500mA	500mA
Raspberry Pi Model A+	700mA	500mA	180mA
Raspberry Pi Model B+	1.8A	600mA/1.2A (switchable)	330mA
Raspberry Pi 2 Model B	1.8A	600mA/1.2A (switchable)	350mA
Raspberry Pi 3 Model B	2.5A	1.2A	400mA
Raspberry Pi 3 Model A+	2.5A	Limited by PSU, board, and connector ratings only.	350mA
Raspberry Pi 3 Model B+	2.5A	1.2A	500mA

Figura 4.4. Comparació del consum energètic entre els diferents models de Raspberry. [5]

Tal i com s'observa en la figura 4.4 entre els models Pi2B, Pi3B i Pi3B+ els que tenen menor consum en general són el model Pi2B i Pi3B, per tant, com ja s'ha comentat, es descarta el model Pi3B+ pel seu consum excessiu que disminuiria el temps de funcionament del sistema de visió submarina.

- **Cost econòmic:** En qualsevol disseny és important tenir en compte el cost dels elements que conformen el dispositiu, ja que quan major és el cost major seria el preu de venda en cas de comercialització i, per tant, major probabilitat de tenir competència.
  En aquest cas, no obstant això, tant el model Pi2B com el model Pi3B tenen preus molt similars, sent generalment el model Pi2B més econòmic, amb un preu mig de 31,90 €.

Finalment, a partir dels diferents factors mencionats anteriorment es decideix seleccionar la Raspberry Pi 2 model B, tot i que també es podria haver seleccionat la Raspberry Pi 3 model B.

L'alimentació d'aquesta computadora de placa reduïda es fa a través d'un connector Micro-USB el qual ha d'assegurar una tensió mínima de 5 V DC amb un amperatge mínim entre 1-2 A i recomanable entre 2,5 i 3 A, si aquesta requereix de la utilització de molts perifèrics, així com un elevat estrès de funcionament de la CPU

4.2.2. Càmera

La càmera utilitzada per capturar les imatges subaquàtiques del sistema de visió submarina està altament condicionada pel tipus de microcontrolador/mini-ordinador seleccionat. En aquest cas, al haver seleccionat la Raspberry Pi 2 model B, l'opció més senzilla, econòmica i més adequada és la utilització del mòdul de càmera que disposa la Raspberry Pi.

Existeixen actualment dos models de mòduls de càmera disponibles i compatibles amb la Raspberry Pi 2 model B: la "camera module v1" i la "camera module v2". La diferència principal entre aquests dos models resideix principalment en la resolució de la càmera: en el primer model és de 5 Megapíxels i en el segon model és de 8 Megapíxels. A més, la diferència de preus és mínima i per tant l'opció més adequada entre els dos models és el de la "camera module v2" amb un preu mig de 23,09 €.

Figura 4.5. Mòdul de càmera V2 connectada amb la Raspberry Pi 2 model B.

Una altra opció, en cas de prescindir del mòdul d'expansió de la Raspberry Pi seria la utilització d'una Webcam amb connexió USB. Cal mencionar que aquesta segona opció no és gaire recomanable ja que, tot i tenir preus relativament més assequibles, la resolució és molt pitjor en comparació a la de la Raspberry Pi, i a més és més difícil tant de configurar com d'utilitzar.

Per tant, per aquest projecte, degut a la seva relació qualitat/preu, resolució d'imatge, facilitat en la seva configuració, així com en la seva utilització, la càmera seleccionada és el mòdul de càmera V2.

4.2.3. Targeta microSD

Per tal de poder instal·lar el sistema operatiu de la Raspberry Pi que executarà el programa que permetrà la captura i el processament de les imatges, així com la comunicació amb el mòdem acústic és necessari l'adquisició d'una targeta de memòria. El model de la Raspberry Pi seleccionada té incorporat un lector de targetes microSD que permet aquesta operació.

Existeixen una varietat molt gran de marques que ofereixen aquest tipus de producte com ara: SanDisk, Kingston, Transcend, Samsung, Verbatim, etc. Per aquest projecte es selecciona la marca Samsung, la qual disposa d'una amplia gamma de models de targetes microSD.

El model EVO Plus de Samsung és un dels millors quant a targetes microSD. Són targetes de memòria de classe 10, les quals asseguren una velocitat de transferència mínima de 10 MB/s. Dintre d'aquest model existeixen diferents targetes en funció de la capacitat de memòria de la qual disposen: 32 GB (14,99 €), 64 GB (24,99 €), 128 GB (48,99), 256 GB (67,99 €) i 512 GB (176,99 €).

Per l'aplicació requerida una targeta de 32 GB ja és suficient, a més la diferència de preus entre les diferents targetes és significativa.

Figura 4.6. Targeta de memòria microSD SAMSUNG EVO Plus 32 GB.

4.2.4. Mòdems acústics submarins

Els mòdems acústics subaquàtics utilitzats són el model S2CM 18/34 de l'empresa EvoLogics, els quals eren un condicionant d'aquest projecte ja que són els que la Universitat tenia disponibles.

Aquest mòdem acústic és sens dubte un dels millors per aplicacions a curtes-mitges distàncies, ja que com s'observa en la taula 3.4 (en la secció de marc teòric) disposa d'una elevada freqüència de l'ona portadora, així com una molt elevada velocitat de transmissió de dades fins a 13900 bits/segon. A més, una de les característiques fonamentals d'aquest mòdem acústic és el seu baix consum energètic configurat de tant sols 0,8 W en recepció de dades i 2,8 W en transmissió de dades a distàncies per sota dels 1000 metres. Aquesta característica el fa ideal pel projecte ja que, com ja s'ha mencionat anteriorment, un dels requeriments indispensables és aconseguir el menor consum energètic possible.

Per altra banda, l'empresa EvoLogics disposa d'una tècnica de modulació única, anomenada S2C, basada en la modulació de senyals per espectres dispersos, la qual permet obtenir majors rendiments en la transmissió de dades, així com major eficiència en entorns submarins de difícil transmissió d'informació.

El model S2CM 18/34 es subministra amb tres possibles interfícies per la seva connexió al microordinador: Ethernet, RS-232 i RS-485. Els models que incorporen les interfícies RS-232 o RS-485 poden utilitzar un mòdul anomenat "Wake-UP" que permet posar els mòdems en un estat de repòs, minimitzant d'aquesta manera el consum energètic del mateix fins a tan sols 2,5 mW. Tanmateix, els mòdems disponibles del laboratori no disposen d'aquest mòdul instal·lat i tenen el connector configurat per utilitzar el protocol de comunicacions TCP/IP sobre Ethernet.

Figura 4.7. Mòdem acústic S2CM 18/34.

4.2.5. Bateries d'alimentació

Prèviament a la selecció de les bateries cal realitzar un estudi de consums energètics teòric, per tal de conèixer la capacitat necessària de les bateries perquè el sistema pugui estar en funcionament el màxim temps possible.

La Raspberry Pi 2 model B té una alimentació de 5 V i els mòdems acústics d'Evologics, en concret el model seleccionat, S2CM 18/34 necessiten una alimentació de 24 V.

Mitjançant el datasheet del fabricant [5] i informació addicional, la Raspberry Pi 2 model B consumeix de sèrie 200 – 230 mA (1-1,2 W), en els quals se li ha d'afegir uns 40 mA a causa de la connexió Ethernet que es requerirà per connectar la raspberry junt amb el mòdem acústic [5]. A més, tot i que no es pot conèixer el valor exacte que consumirà la CPU pel programa que s'executarà (sense mesurar-ho amb un amperímetre directament) el fabricant estableix que amb tasques computacionals molt elevades es sumen 200 - 250 mA més, per tant en aquest projecte es podrien considerar de 150 mA. En total un consum energètic de 390 ≈ 400 mA (2 W).

El consum de la càmera seleccionada s'estableix com a màxim en uns 250 mA (1,25 W) en constant funcionament. En el cas real aquest consum serà molt inferior, tot i així pels càlculs teòrics es considerarà el cas més desfavorable.

El consum energètic dels mòdems acústics subaquàtics seleccionats, segons en el datasheet de la empresa Evologics [6] es troba distribuït segons 4 modes de funcionament diferents:

-**Stand-by Mode:** mode de repòs amb un consum de 2,5 mW.

-Listen Mode: mode exclusiu per connexions amb RS-232 (per tant es descarta pel projecte).

-Receive Mode: mode de captació de dades amb un consum inferior a 0,8 W.

-Transmit Mode: mode de transmissió de dades amb un consum variable segons la distància entre els mòdems. Pel projecte amb un rang de 1000 m ja n'hi ha suficient, el consum és aproximadament 2,8 W.

Power Consumption		
	Stand-by Mode	2.5 mW
	Receive Mode	less than 0.8 W
	Transmit Mode	2.8 W, 1000 m range 8 W, 2000 m range 35 W, 3500 m range 55 W, max. available
Power Supply		
	External	24 VDC (12 VDC, 300 VDC options)

Figura 4.8. Consums energètics del mòdem acústic S2CM 18/34. [6]

Per tal de poder fer suposicions de consum energètic pel mòdem que estarà submergit, es pot considerar el cas més desfavorable en què aquest transmetrà dades constantment. Per tant un consum de 2,8 W que equival a 2,8 W /24 V ≈ 120 mA.

Consum energètic total: 2 W (Raspberry) + 1,25 W (Càmera) + 2,8 W (mòdem) = 6,05 W.

Suposant una capacitat de bateries elevada estàndard per aquest tipus d'aplicacions d'uns 18 AH, el temps que podrà estar en funcionament el sistema dissenyat sota l'aigua seria de:

$$temps = \frac{Voltatge\ Bateries\ (V) * Capacitat\ bateria\ sèrie\ (A*Hora)}{consum\ energètic\ (W)} = \frac{29{,}6\ V \cdot 18\ AH}{6{,}05\ W} = 87{,}93\ h = 3{,}66\ \text{dies} \qquad (1)$$

Aquest temps seria suposant el consum energètic més elevat que es podria esperar tenint en compte els casos més desfavorables, i capturant imatges i enviant-les durant els 3,66 dies.

També és necessari destacar que aquest càlculs de consums energètics són teòrics i amb les informacions obtingudes de datasheets. Per tant, amb resultats experimentals és molt probable que els valors siguin inferiors i en conseqüència el temps durant el qual es podria disposar de la Raspberry i el mòdem en funcionament podria ser major.

Un cop realitzat l'estudi energètic teòric, per alimentar el mòdem acústic caldrà seleccionar una bateria d'una capacitat aproximada de 18 AH que permeti subministrar 24 V i que tingui una mida ajustada. Degut a què l'empresa BlueRobotics subministra bateries i altres productes destinats a dispositius

aquàtics, així com tubs a mida per l'encapsulament d'aquests dispositius, s'opta per seleccionar dues bateries d'ions de liti de 14,8 V del mateix fabricant, per així connectar-les en sèrie i disposar de 29,6 V, suficients per alimentar tant el mòdem com la Raspberry Pi, fent servir dos reguladors reductors de tensió DC-DC.

Figura 4.9. Bateria de 14,8 V – 18 AH. [8]

Les bateries també es podrien comprar d'un altre fabricant les quals oferissin, sense necessitat d'un regulador de tensió, els 24 V necessaris per l'alimentació dels mòdems acústics subaquàtics. Tot i així, es seleccionen les del mateix fabricant, ja que són molt petites en relació a altres que subministren la mateixa tensió i capacitat, tenen la mida exacta per als tubs d'encapsulament, i generalment trobar bateries amb unes mides determinades sol ser molt complicat i en certes ocasions pràcticament impossible. A més, a nivell de costos, l'estalvi econòmic que suposaria eliminar el regulador de tensió i prescindir d'una de les bateries tampoc és gaire significatiu, ja que el preu d'una bateria de 24 V és semblant a la suma de les dues bateries de 14,8 V les quals tenen un preu de 221,56 € cadascuna.

4.2.6. Reguladors de tensió DC-DC

Per tal d'alimentar la Raspberry Pi model 2 B a 5 V i el mòdem acústic transmissor S2CM 18/34 a 24 V es necessiten dos reguladors reductors de tensió DC-DC, ja que les bateries seleccionades en sèrie entreguen una tensió de 29,6 V.

Existeixen principalment dos tipus de reguladors reductors de tensió DC-DC:

- **Reguladors reductors DC-DC lineals:** es tracta de reguladors que controlen la tensió de sortida ajustant contínuament la caiguda de tensió en un transistor de potència connectat en sèrie entre la entrada no regulada i la càrrega. El transistor ha de conduir corrent contínuament i per tant opera en la seva regió activa o lineal. Aquests dispositius són generalment més econòmics i senzills d'utilitzar, però tenen l'inconvenient que al disposar d'un transistor treballant contínuament en la seva regió lineal tenen més pèrdues energètiques i per tant són poc eficients, arribant a percentatges de tant sols un 20% d'eficiència energètica.
- **Reguladors reductors DC-DC commutats:** el reguladors commutats utilitzen un transistor de potència com a commutador d'alta freqüència, de tal manera que l'energia es transmet des

de l'entrada a la càrrega en diversos paquets discrets, obtenint una ona quadrada. Els polsos d'intensitat es converteixen posteriorment en corrent continua mitjançant filtres inductius i capacitius. En aquest cas, el transistor treballa en les regions de tall (actuant com a un interruptor obert) i en la de saturació (actuant com a un interruptor tancat) obtenint majors rendiments energètics, degut a què aquest consumeix menys potència que treballant en la seva regió activa o lineal. Aquest tipus de reguladors permeten obtenir alts rendiments energètics, al voltant d'un 80-90%, però amb circuits electrònics més complexos.

Ja que el sistema a dissenyar en aquest projecte requereix de mínimes pèrdues d'energia és necessari utilitzar un regulador Step-Down o Buck, per tal d'obtenir així el màxim rendiment energètic possible.

El regulador escollit és el Step-Down LM2596 el qual suporta corrents de sortida fins a 3 A, permet voltatges d'entrada d'entre 4,5 V fins a 40 V i proporciona tensions de sortida d'entre 1,23 V fins a 37 V regulables mitjançant un potenciòmetre multivolta. S'ha seleccionat aquest model ja que disposa d'un alt rendiment energètic (70-90%), té un baix voltatge d'arrissada i disposa d'una freqüència de commutació molt elevada de 150 kHz, característica essencial pel sistema, per tal que no interfereixi amb la freqüència de funcionament dels mòdems acústics (18-34 kHz). El seu preu és de 1,45 €.

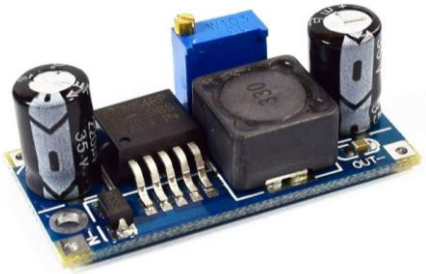

Figura 4.10. Regulador Step-Down LM2596. [9]

4.2.7. Encapsulament

L'encapsulament del sistema de visió submarina és un aspecte molt important a tenir en compte, ja que pot dependre d'aquest factor el correcte funcionament del sistema.

Generalment hi ha dos mètodes bàsics d'encapsulament per aplicacions submarines: utilitzar resina per recobrir tot el sistema, per tal d'evitar el contacte dels components electrònics amb l'aigua i per suportar les pressions subaquàtiques que podrien danyar els components, i per altra banda utilitzar tubs hermètics per tal de protegir els dispositius electrònics.

Ambdues opcions disposen d'avantatges i inconvenients les quals s'han de tenir en compte alhora de seleccionar l'opció adient pel projecte. Utilitzar resina és una opció econòmica i força segura que

protegeix els dispositius electrònics, permetent a l'usuari utilitzar una major o menor quantitat de resina en funció de la profunditat requerida per l'aplicació seleccionada. L'inconvenient principal de la resina és que un cop s'ha fixat el sistema, recuperar-lo és molt difícil i generalment queda clausurat per sempre.

Per altra banda, utilitzar tubs hermètics té l'avantatge que es pot recuperar el sistema en qualsevol moment. No obstant això, el seu preu es molt més elevat i les opcions són fixes, és a dir, s'ha de seleccionar un tub prèviament dissenyat que suporti profunditats establertes pel fabricant.

Per aquest projecte l'opció del tub hermètic és la més indicada, ja que tot i tenir un preu molt més elevat que la resina, és essencial recuperar el sistema per tal d'extreure les fotografies capturades per la Raspberry Pi que no s'han pogut enviar correctament, així com implementar modificacions en la programació segons les necessitats del usuari. A més, la càmera ha de poder estar encarada en un espai transparent del tub per tal d'obtenir fotografies nítides, un motiu més pel qual la resina no és una opció adient per aquesta aplicació.

En l'encapsulament tant sols s'inclourà els cables del mòdem acústic transmissor, les bateries, els reguladors de tensió i la Raspberry Pi amb la càmera. El mòdem acústic receptor i l'ordinador estaran a l'exterior directament connectats a una font d'alimentació externa, tal i com es pot apreciar en l'esquema del disseny previ (veure apartat 4.1 Esquema previ).

Tal i com s'ha mencionat anteriorment, s'utilitzarà el mateix fabricant de les bateries per l'encapsulament, per tal d'assegurar que aquestes disposin de les mides exactes. Així doncs, existeixen dos possibles opcions d'encapsulament pel projecte:

- **Opció 1, un únic tub:** A través de la pàgina web del fabricant "BlueRobotics" es selecciona un tub acrílic de 6'' (15,2 cm) de diàmetre i una llargada de 29,8 cm, amb capacitat de suportar profunditats d'aproximadament 100 m. Les dues bateries de 14,8 V tenen una llargada de 14,1 cm i 7,5 cm de diàmetre, suficients per disposar de l'espai necessari per col·locar la Raspberry Pi model 2 B, la càmera i els dos reguladors de tensió. Per tal que la càmera pugui capturar les fotografies es selecciona, dins les opcions que aquest tub disposa, una tapa acrílica transparent on anirà encarada la càmera. També, per tal de protegir els cables d'alimentació del mòdem acústic es selecciona una tapa d'alumini amb diversos forats on aniran col·locats. El preu final del tub hermètic és de 290 $.

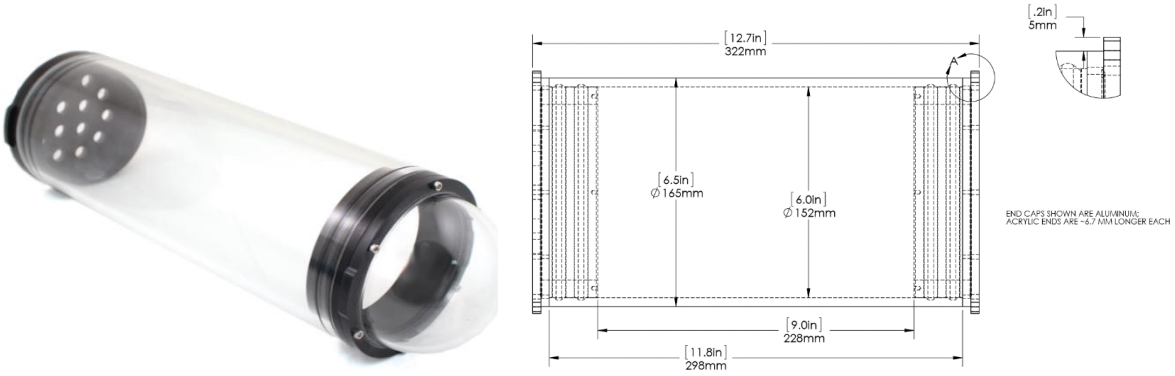

Figura 4.11. Tub hermètic d'encapsulament pel sistema de visió submarina (opció 1). [10]

- **Opció 2, dos tubs:** Igual que en l'opció 1 s'utilitza el fabricant "BlueRobotics" i es seleccionen dos tubs acrílics, un de 3'' de diàmetre i una llargada de 29,8 cm i un altre de 2'' de diàmetre amb la mateixa llargada. En el de 3'' es col·loquen les dues bateries de 14,8 V, ocupant així tot l'espai del mateix i en l'altre tub es col·loca la Raspberry Pi 2 model B, així com dels reguladors de tensió i de la càmera. El preu dels dos tubs és de: 110+143$ =253 $.

Tot i ser més econòmica l'opció de dos tubs hermètics, la configuració resulta molt tediosa degut a què el sistema requeriria de connectar els cables d'alimentació de les bateries en l'altre tub on hi hauria tant el mòdem acústic amb els cables connectats en l'interior del mateix com la Raspberry Pi, els reguladors i la càmera. Per tant, degut a què l'estalvi econòmic no és suficientment significatiu i resulta ser una opció molt més pràctica disposar tot el sistema en un sol encapsulament es selecciona la primera opció.

5. Alimentació i configuració del sistema dissenyat

5.1. Cablejat i alimentació

Per tal d'alimentar el mòdem acústic S2CM 18/34, aquest disposa d'un connector anomenat "Subconn Metal Shell FCR1508M"el qual es pot observar en la figura 5.1, especialment dissenyat per aplicacions on es requereixen una connectivitat sota l'aigua molt resistent i altament protegida.

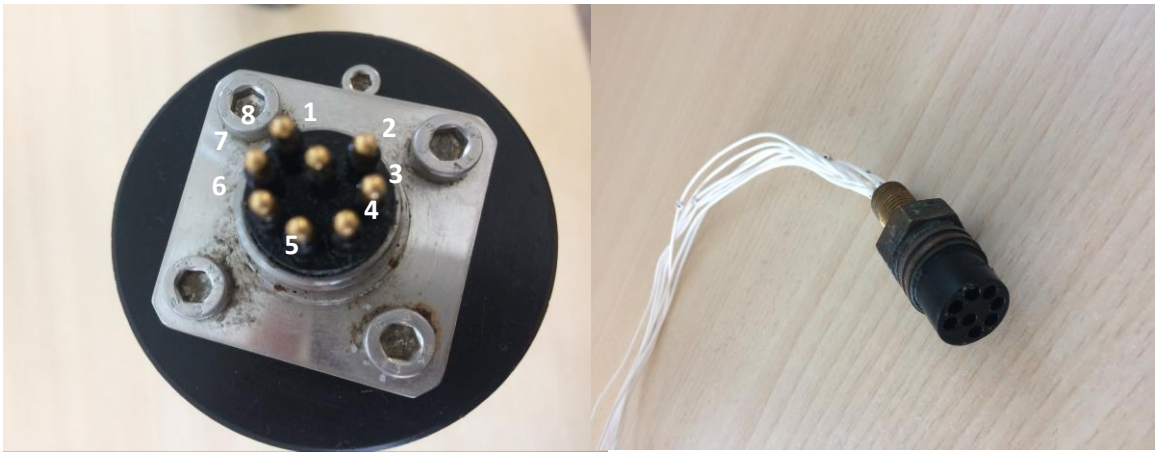

Figura 5.1. Connector "Subconn Metal Shell FCR1508M".

El cablejat s'ha de realitzar, ja que no ve inclòs amb el mòdem acústic. Per tant a partir del pin-out del mòdem, s'ha de cablejar el connector femella amb l'alimentació i les connexions Ethernet.

Taula 5.1. Pin-out de configuració del connector del mòdem acústic.

Descripció	PIN
TxD+ OUT	1
NC	2
POWER ON	3
V- (Power GND)	4
V+ (24 V)	5
RxD- IN	6
RxD+ IN	7
TxD- OUT	8

Per aquest tipus de connexió es requereix d'un cable UTP amb connector RJ45 i un cablejat Ethernet creuat, ja que s'està interconnectant dos equips directament entre ells, els quals es comunicaran alternativament.

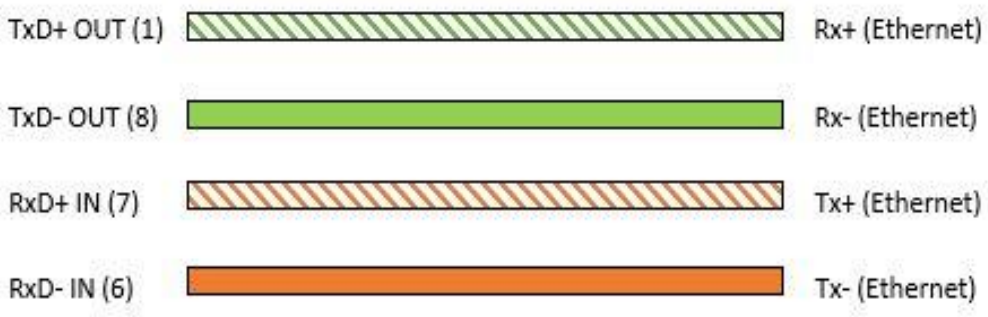

Figura 5.2. Connexió Ethernet creuada entre mòdem acústic i la Raspberry Pi.

A partir de les bateries i els reguladors seleccionats en el capítol anterior (veure capítol 4. Selecció dels components del sistema de visió submarina), es connecten les dues bateries en sèrie, per tal d'obtenir la suficient tensió com per alimentar al mòdem acústic transmissor i posteriorment es redueix la tensió mitjançant el regulador Step-down LM2596 a 5 V per alimentar la Raspberry Pi model 2 B i a 24 V respectivament per alimentar el mòdem acústic S2CM 18/34 transmissor, l'esquema es pot observar en la figura 5.3.

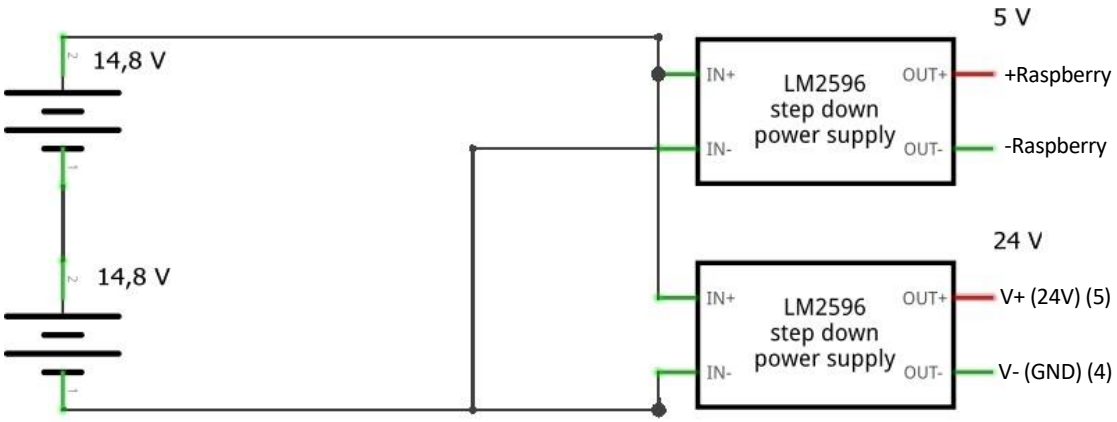

Figura 5.3. Esquema d'alimentació del mòdem acústic i la Raspberry Pi.

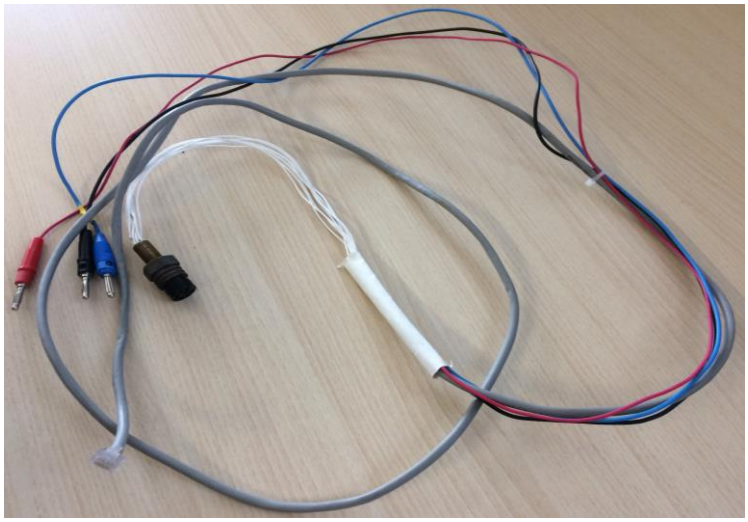

Figura 5.4. Connector per l'alimentació i comunicació Ethernet del mòdem acústic.

Tal i com es mostra en la figura 5.4, el connector per alimentar i comunicar el mòdem acústic amb la Raspberry Pi queda cablejat finalment amb un cable Ethernet, amb les connexions mencionades en la figura 5.2, i tres connectors tipo banana: el vermell corresponent a l'alimentació positiva V+ 24 V (5), el negre corresponent a l'alimentació negativa V- (4) i finalment el blau corresponent al POWER ON (3), el qual s'ha de connectar al negatiu per tal de què el mòdem estigui operatiu.

5.2. Configuració Raspberry Pi 2 model B, mòdul de càmera V2 i PC

Prèviament a la programació de la Raspberry Pi és necessari la instal·lació del sistema operatiu de la mateixa. Existeixen diferents sistemes operatius oficials compatibles amb aquesta computadora de placa reduïda: Raspbian, Raspbian Lite (versió reduïda de Raspbian, sense escriptori gràfic i per tant sense ninguna aplicació que el requereixi com ara Wolfram Engenie, Mathematic, LibreOffice, etc), OSMC, LibreELEC, Ubuntu Snappy Core (servidor sense escriptori gràfic), Ubuntu MATE i RISC OS.

Entre tots ells Raspberry recomana la instal·lació de Raspbian, ja que es tracta d'un sistema operatiu de tipus GNU/Linux amb una interfície gràfica senzilla i pròxima a l'usuari.

Per dur a terme la instal·lació del sistema operatiu, Raspberry ofereix un instal·lador anomenat "NOOBS" el qual facilita el procés. Un cop s'ha instal·lat és necessari formatejar la targeta microSD, i a continuació extreure els fitxers en la mateixa. Posteriorment es reinicia la Raspberry (desconnectant i connectant l'alimentació) i comença la instal·lació.

Per tal de poder realitzar la instal·lació és necessari disposar d'un monitor, un cable HDMI (per la connexió entre el monitor i la Raspberry Pi), un ratolí i un teclat. A continuació, es selecciona l'opció "Raspbian" en l'instal·lador i s'instal·la el sistema operatiu. Finalment, tal i com s'observa en la figura

5.5 apareix l'escriptori gràfic de Raspbian indicant que la instal·lació ha finalitzat i s'ha dut a terme correctament.

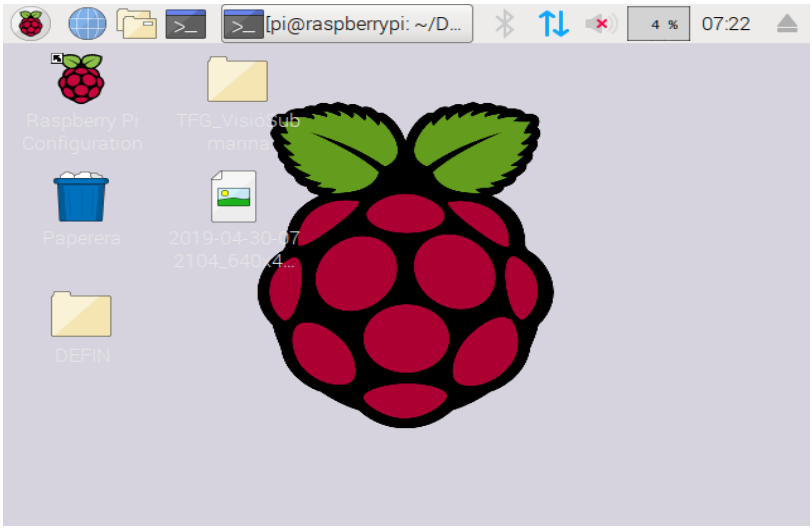

Figura 5.5. Escriptori gràfic del sistema operatiu Raspbian.

Posteriorment és necessari modificar la data, ja que el programa que s'executarà requereix tenir-la actualitzada, per guardar el dia en la que s'han capturat les fotografies. Per dur a terme la modificació es pot utilitzar la consola de Raspbian i introduir el següent comandament: "sudo raspi-config". A partir d'aquest comandament es selecciona l'opció "Internationalization options", "Change Time Zone" i es canvia l'àrea geogràfica a Europa i es selecciona la ciutat/regió de Madrid. Finalment, a partir del comandament "sudo 05092027" es configura el mes, el dia i l'hora, en l'exemple concret el mes de maig, el dia 9 i l'hora 20:27.

L'habilitació de la càmera per tal de poder utilitzar-la, es duu a terme a través del panel de configuració de la Raspberry Pi. El comandament per accedir-hi és "sudo raspi-config". Un cop s'accedeix, en la configuració es selecciona l'opció "Interfacing Options" i finalment l'opció "P1 Camera", tal i com s'observa en la figura 5.6, on s'habilita el mòdul de càmera de la Raspberry Pi.

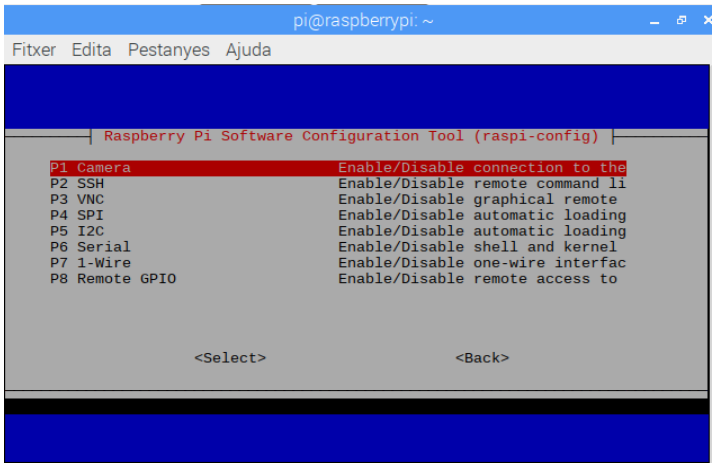

Figura 5.6. Habilitació del mòdul de càmera de la Raspberry Pi.

Per tal d'utilitzar la càmera a través de Python és necessari instal·lar una llibreria mitjançant el comandament "sudo apt-get install python-picamera"

A continuació, és necessari modificar l'adreça IP de la Raspberry Pi a una adreça estàtica per tal que sigui compatible amb la del mòdem acústic transmissor. Per tal de poder enllaçar la Raspberry Pi i el mòdem acústic, i comunicar-se a través del protocol de comunicacions TCP/IP sobre Ethernet, cal que aquests compleixin una sèrie de requisits.

Per començar, els dos disposen d'una direcció MAC (*Media Acces Control)* que ve fixada pel hardware dels dispositius. Aquesta adreça està formada per 48 bits i serveix per identificar de forma única a un dispositiu. A continuació, aquests han de disposar d'una adreça IP única, així com d'una màscara de subxarxa, que determina amb quins dispositius es pot comunicar o no. A més, per tal d'enllaçar els dos dispositius, aquests han de disposar d'una direcció IP privada de la mateixa classe. Existeixen tres tipus de classes de direccions IP privades: classe A, de 10.0.0.0 fins a 10.255.255.255, per grans xarxes privades, classe B, de 172.16.0.0 fins a 172.31.255.255, per a xarxes mitjanes, i finalment classe C, de 192.168.0.0 fins a 192.168.255.255, per a xarxes petites i comunicacions entre dispositius.

El mòdem acústic transmissor i la Raspberry Pi s'han configurat amb direccionament de classe C: el módem ja tenia configurada l'adreça 192.168.1.196 i la Raspberry Pi per tant haurà de ser dins del mateix rang, 192.168.1.X.

Per tal de canviar una IP dinàmica a una estàtica es requereix d'accedir a la configuració d'interfícies de xarxes mitjançant el comandament "sudo nano /etc/dhcpcd.conf". A partir dels comandaments que es poden observar en la figura 5.7 es configura la IP estàtica que es desitja, així com el nom de la connexió ethernet. En aquest cas particular la interfície és la enxb827ebe504ff i l'adreça la 192.168.1.8.

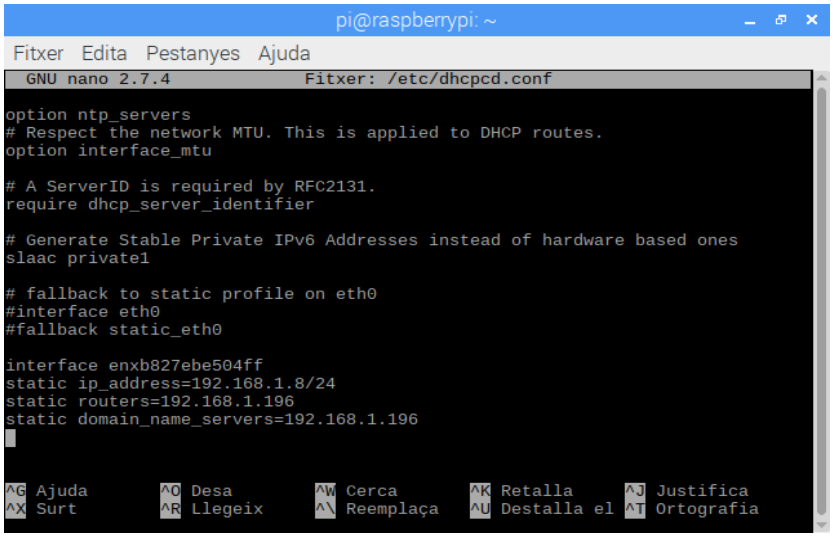

Figura 5.7. Configuració IP estàtica de la Raspberry Pi .

De la mateixa manera que s'ha modificat la IP de la Raspberry Pi per tal de què aquesta sigui compatible amb la del mòdem acústic transmissor, és necessari configurar la IP del PC/Portàtil que rebrà les imatges del mòdem receptor. Per dur a terme aquesta modificació en Windows, cal accedir al tauler de control i dintre de les configuracions de l'adaptador Ethernet, modificar la IPV4 d'una IP dinàmica a una IP estàtica de classe C, tal i com s'observa en la figura 5.8, establint la seva direcció IP en 192.168.1.50.

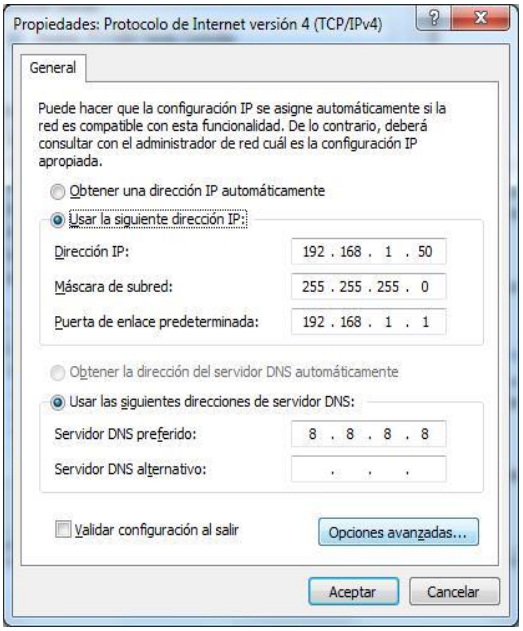

Figura 5.8. Configuració de la direcció IP estàtica del PC receptor d'imatges.

Per finalitzar amb la configuració de la Raspberry Pi és necessari que el programa s'executi automàticament en alimentar la placa, sense necessitat de connectar-la a un monitor amb teclat i ratolí i que un usuari l'executi manualment. Aquest concepte s'anomena "Autoboot", el qual fa referència a executar automàticament un programa quan la Raspberry s'inicialitza per primer cop.

A partir del comandament "sudo nano /etc/rc.local" s'accedeix a un fitxer de text editable amb el qual s'indica al software del programa el directori en el qual es troba i el nom del mateix, tal i com es pot apreciar en la figura. 5.9. El símbol "&" al final de la instrucció, serveix per indicar que es tracta d'un programa el qual s'executarà en un bucle infinit i per tant no acabarà mai. És necessari col·locar el símbol ja que sinó el programa no s'executarà. Finalment, es conclou col·locant la instrucció "exit 0" per indicar el final del fitxer.

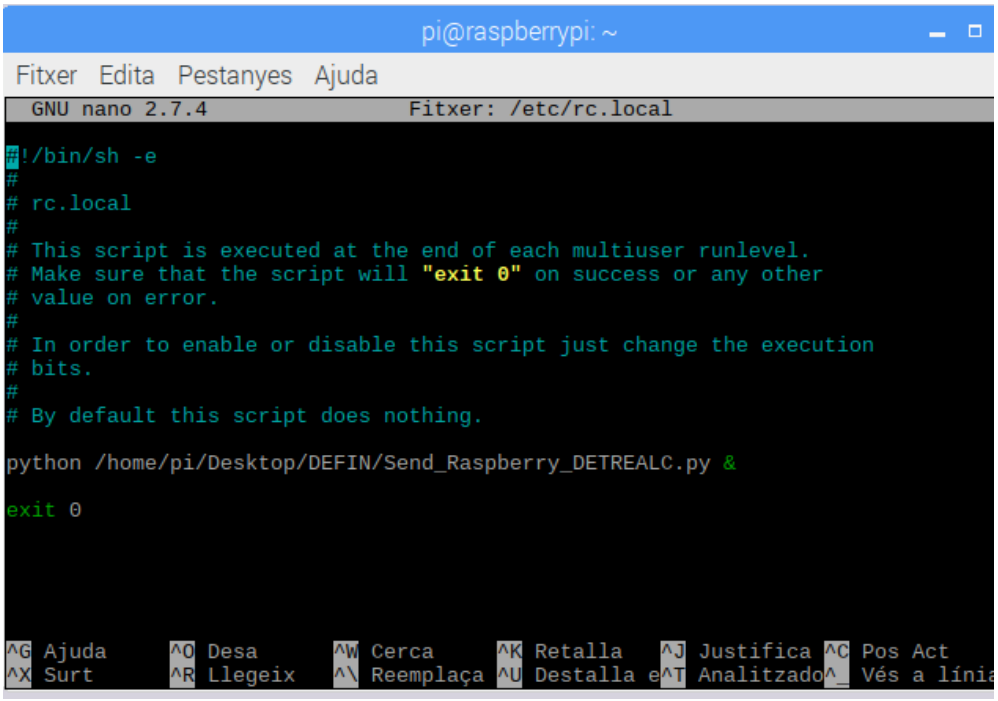

Figura 5.9. Configuració de la inicialització del programa en "Start-up".

5.3. Configuració Mòdems acústics EvoLogics S2CM 18/34

Prèviament a la configuració dels mòdems és necessari introduir el concepte de "AT Commands", ja que aquests dispositius són controlats mitjançant aquest tipus de llenguatge de comandaments. Els comandaments AT són cadenes de text curts que indiquen al dispositiu que realitzi determinades accions. El conjunt d'ordres AT conté comandes de totes les operacions de comunicació, ordres per visualitzar, modificar i desar la configuració del dispositiu, així com ordres de transmissió i posicionament.

Els mòdems acústics d'EvoLogics, en concret el model S2CM 18/34, disposen de dos modes de funcionament quant a les dades entrants:

- **Data Mode (mode de dades):** es tracta d'un mode de funcionament en què totes les dades que rep el mòdem s'envien a través del canal acústic sense excepció ni comprovar res.
- **Command Mode (mode d'ordres):** en aquest mode de funcionament la informació entrant es tracta com a ordres i només es duen a terme en cas de ser reconegudes com a vàlides. Només s'enviaran a través del canal acústic les dades precedides de la comanda "AT*SEND".

Existeixen dos tipus d'interpretadors de comandaments per als mòdems acústics S2C d'EvoLogics. El filtre at, el qual permet els dos tipus de modes de funcionament descrits anteriorment, tant el Data Mode com el Command Mode, i el filtre net, el qual únicament incorpora el mode de funcionament d'ordres.

El filtre at és molt útil per certes aplicacions, tot i que té la particularitat que el canvi entre els dos modes de funcionament és molt tediós. Un dels inconvenients principals del filtre at és que no es té un control de les dades que s'envien quan està seleccionat el Data Mode, és a dir, quan es fa la petició d'enviament de dades no es té control de si s'han enviat realment o no, o si s'han perdut paquets d'informació durant l'enviament.

Per tant, tot i que el mode de funcionament de dades permet enviar un conjunt molt gran de dades de forma molt ràpida, per l'aplicació del projecte no és útil, ja que es necessita tenir un control dels paquets de dades que s'han enviat correctament i dels que no per tal de poder tornar-los a enviar, ja que sinó la imatge resultaria incompleta. Cal mencionar que el filtre at és útil per aplicacions en les que no importa si es perden o no dades, com per exemple en l'enviament de mesures d'un sensor de temperatura, en el que no importa si certes mesures es perden o no.

El filtre net, incorpora a més una instrucció anomenada "AT*SEND" la qual permet enviar paquets de dades de fins a 1024 bytes amb control dels mateixos. Aquesta instrucció és la principal diferència entre els dos filtres, ja que el filtre at incorpora les mateixes instruccions que el filtre net, exceptuant aquesta última. La instrucció "AT*SENDIM" permet també enviar paquets de dades de fins a 64 bytes amb control dels mateixos. No obstant això, aquesta instrucció no és gaire útil pel projecte, ja que a diferència del "AT*SEND", els missatges no s'emmagatzemen a la cua, sinó que s'envien un a un esperant a què el destinatari ho rebi, i en cas d'enviar un altre missatge mentre s'està enviant un anterior, la transmissió es cancel·la. Per tant, per la realització del projecte els mòdems acústics estaran configurats amb el filtre net.

De sèrie els mòdems venen configurats amb el filtre at. Per tant, per tal de poder modificar aquest paràmetre és necessari accedir via protocol http a la configuració web dels mateixos a través de la seva direcció IP, alimentant-los i connectant-los a un portàtil/PC via Ethernet.

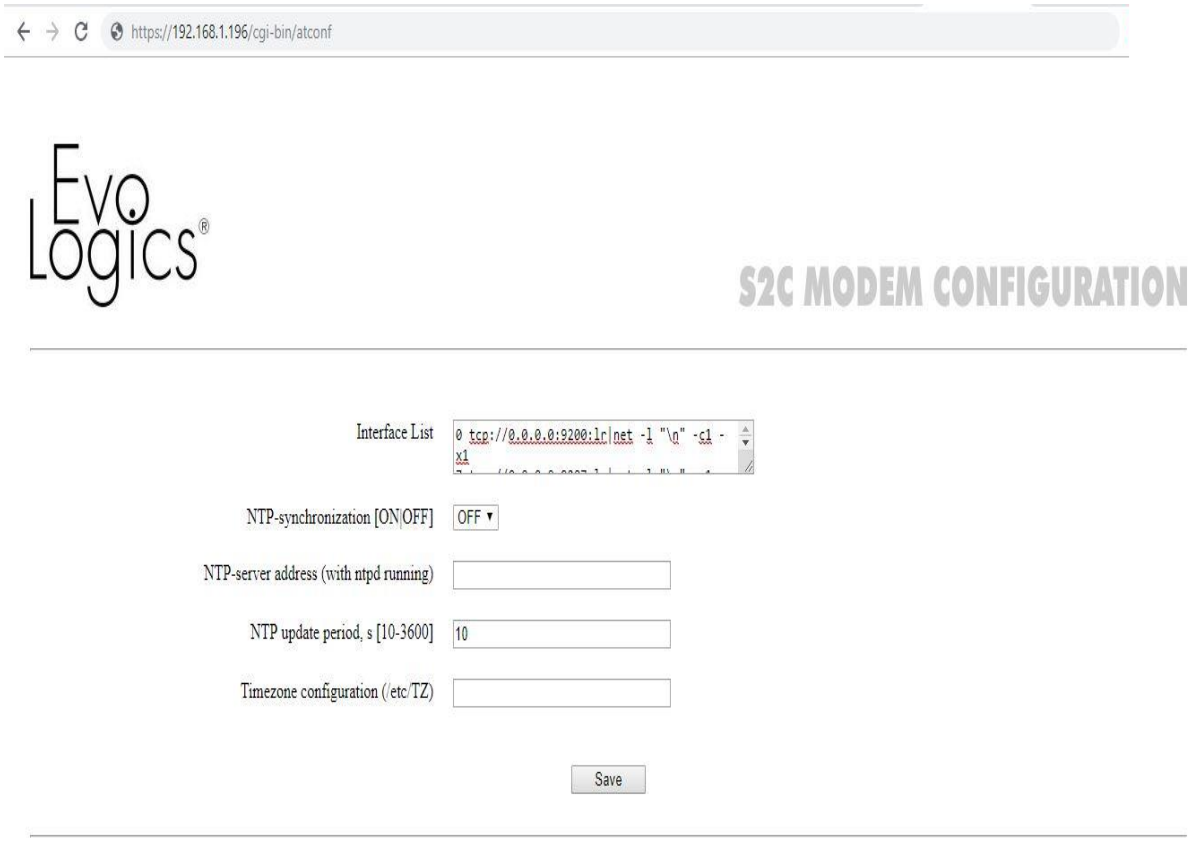

Figura 5.10. Configuració web dels mòdems acústics.

Com s'observa en la figura 5.10 s'ha accedit a la configuració web del mòdem acústic amb la seva adreça IP: 192.168.1.196. A través de l'opció "Interface String List" es pot modificar el filtre, així com certs paràmetres del mòdem acústic:

Interface List -> 192.168.1.196: tcp://0.0.0.0:9200:lr|net -l "\n" -c 1 -x 1

tcp://0.0.0.0:9207:lr|net -l "\n" -c 1 -x 1

Primerament s'indica la direcció IP seguida del protocol de comunicacions utilitzat, el qual com es pot veure en l'exemple d'amunt és TCP. Seguidament s'indica els ports de comunicacions (9200 pel primer canal i 9207 pel segon canal) els quals han de ser els mateixos pels dos mòdems acústics per assegurar la comunicació entre els dos. Posteriorment apareix el tipus de filtre que es vol utilitzar i un conjunt de

paràmetres (els quals prenen dos valors, 1 o 0 en funció de si estan o no habilitats) que permeten modificar certes característiques dels mòdems:

- **-c:** habilitació del control de la configuració global.
- **-x:** activació de les notificacions esteses.
- **-u:** activació de la informació de posicionament del mòdem.
- **-f:** habilitació del mode de protocol estès.

Els mòdems acústics poden trobar-se en tres possibles estats:

- **Listen State:** estat en el que el mòdem acústic es troba preparat per acceptar qualsevol intent d'establiment de comunicació acústica.
- **Noise State:** estat en el que el mòdem acústic no pot establir cap comunicació acústica.
- **Deaf State:** estat en el que el mòdem acústic no pot rebre cap senyal i únicament pot funcionar com a transmissor de dades.

Per tal de què dos mòdems puguin comunicar-se entre ells i ignorar altres senyals procedents d'altres mòdems acústics, aquests disposen d'una adreça local i única. Existeix un mode de protocol estès el qual, si està inhabilitat, els missatges que s'envien del mòdem acústic transmissor només seran rebuts per aquells mòdems que disposin de l'adreça local a la qual estiguin referenciats, així com d'un paràmetre anomenat identificador de canal ID que ha de ser igual tant pels mòdems transmissors com receptors.

Quant a la configuració dels mòdems cal mencionar que existeixen principalment dos tipus de comandaments AT en relació a la configuració de paràmetres, els que són merament informatius i serveixen per informar sobre l'estat de certes característiques dels mòdems (AT?) i els que permeten modificar-les (AT!/AT@).

Els comandaments AT més rellevants per la configuració dels mòdems de cara a la realització del projecte són:

- **ATA:** comandament per modificar l'estat dels mòdems acústics a "Listen State".
- **ATZn:** comandament que en funció de la lletra n, la qual pot prendre 5 possibles valors, pot realitzar diverses funcions: si n val 0 es realitza un reset de fàbrica del dispositiu i es tanquen totes les possibles comunicacions TCP; si n val 1, es tanquen les comunicacions acústiques i s'eliminen les possibles dades que hi haguessin en el buffer del mòdem; el número 2 està reservat per futur ús i no té cap funcionalitat; amb n igual a 3 s'eliminen les possibles peticions d'enviament de missatges instantanis; finalment amb n igual a 4, es neteja el buffer de transmissió.

- **AT?AL/AT!ALn:** serveixen per obtenir l'adreça local del mòdem acústic, així com per poder modificar-la. El mòdem acústic transmissor disposa de l'adreça local 2 i el receptor de la 3.
- **AT?RC/AT!RCn:** comandaments per mostrar i modificar el número d'intents de restabliment de connexions acústiques. El valor n es redueix fins que el mòdem ha realitzat el número d'intents establerts. Quan arriba a 0 les següents peticions d'enviament de dades tant sols es realitzaran un únic cop fins que la comunicació acústica resulti exitosa, on el valor n prendrà el valor original. Els mòdems han estat configurats amb un valor de n de 3 (AT!RC3), recomanat pel fabricant.
- **AT?ZF/AT@ZFn:** comandaments per mostrar i modificar l'estat del mode de protocol estès. Per aquesta aplicació ha estat inhabilitat (AT@ZF0) per tal d'assegurar l'enviament i la recepció de les dades entre els mòdems acústics transmissor i receptor.
- **AT?ZI/AT!ZIn:** defineix el temps en el que una connexió acústica serà tancada en cas de què el mòdem no rebi ni transmeti informació. Aquest paràmetre ha estat configurat amb un temps de 120 segons (AT!ZI120), ja que és el recomanat pel fabricant i el que ve de sèrie configurat amb els mòdems.
- **AT?ZX/AT@ZXn:** comandaments per mostrar i modificar el control de les notificacions esteses en l'enviament de dades. Quan aquestes són habilitades aporten informació de posicionament del mòdem, duració i velocitats de transmissió, així com informació sobre el retràs en els enviaments. Són informacions potencialment útils però no es tenen en consideració en el projecte ja que retarden el temps d'enviament de les fotografies degut a què són moltes dades d'informació que s'envien en conjunt amb les dades d'interès. De totes maneres, tot i que s'han desactivat amb el comandament AT@ZX0, el programa també funcionaria correctament si fossin habilitades.
- **AT?ZL/AT@ZLn:** defineix la mida en bytes del buffer de transmissió dels mòdems acústics. Aquest pot tenir una mida d'entre 8096 – 2097152 bytes. Degut a què la mida de les fotografies pot variar en funció de la resolució i el grau de compressió de la imatge, s'ha seleccionat la mida més gran possible del buffer per assegurar que no es desbordi en cas de l'enviament d'una imatge molt gran.
- **AT?S:** aquest comandament serveix per obtenir informació de l'estat del mòdem. Retorna l'adreça local del mòdem acústic, l'adreça remota (si ha estat prèviament configurada), l'estat en el que es troba el mòdem, el número de paquets de dades que estan programats per ser enviats, el número de bytes del que disposa el buffer del mòdem i informació sobre l'anomenat "Promiscuous Mode" el qual fa referència si aquest rebrà informació de qualsevol mòdem que estigui enviant o només d'aquell en el qual estigui adreçat.

Per comunicar-se amb els mòdems acústics es pot establir una comunicació TCP/IP obrint un socket entre la Raspberry Pi i el mòdem acústic mitjançant el llenguatge de programació seleccionat (veure capítol 6. Programació (Software)). Tanmateix, per configuracions i instruccions senzilles també es pot fer servir el programari Netcat que és més senzill i ràpid.

Netcat permet a través d'un intèrpret de comandaments i una sintaxis senzilla obrir ports TCP/UDP en un host, quedant Netcat a l'escolta. El principi de funcionament és el mateix que es realitza en el primer mètode anteriorment explicat. S'obre un socket per connectar-se al mòdem acústic, s'envia tot el que arriba per l'entrada estàndard a través del socket i finalment es treu tot el que ha rebut el socket destí per la sortida.

Per accedir al mòdem acústic, primerament s'ha d'alimentar i connectar-lo a un PC/portàtil a través d'Ethernet. A continuació, s'obre Netcat i s'introdueix la direcció IP del mòdem seguida del port al qual es vol establir la connexió. Un cop s'ha connectat, a partir de comandaments AT es pot interactuar amb el mòdem, tal i com es mostra en la figura 5.11.

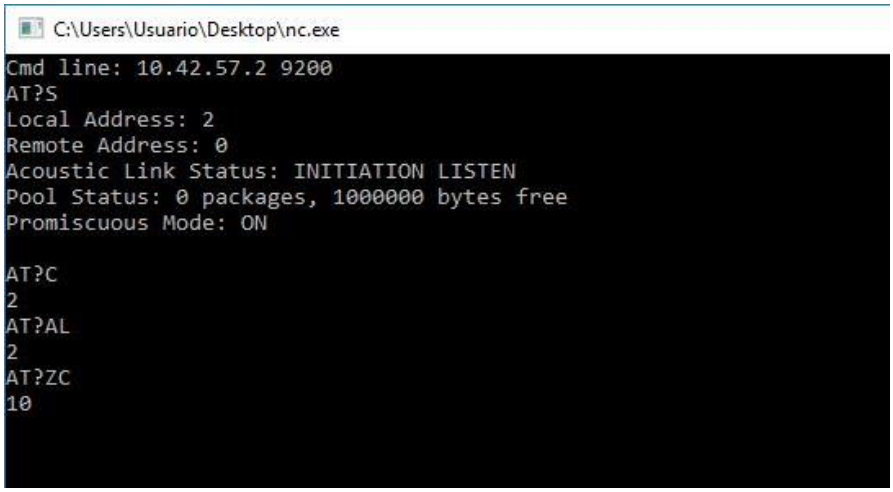

Figura 5.11. Configuració dels mòdems acústics mitjançant Netcat.

6. Programació (Software)

Aquest capítol té la intenció d'explicar el software del sistema de visió submarina, introduint tant l'ús de l'emulador de mòdems acústics virtuals de l'empresa EvoLogics, com incidint en els diferents blocs funcionals dels programes, així com detallant les característiques i peculiaritats dels mateixos.

L'objectiu inicial del projecte era la realització d'un software capaç de capturar i enviar el major nombre d'imatges de forma automàtica en el mínim temps possible. No obstant això, a mesura que es va avançar amb el projecte es va realitzar una segona versió del software que enviés imatges de forma selectiva i configurable per l'usuari.

6.1. Llenguatge de programació Python

6.1.1. Selecció del llenguatge i peculiaritats

El llenguatge de programació seleccionat per tal de programar la Raspberry Pi, així com l'ordinador que actuarà com receptor de les imatges, és Python. S'ha elegit aquest llenguatge de programació d'alt nivell per diversos motius:

- **Àmplia documentació:** Python és un llenguatge de programació que disposa d'una extensa documentació, amb molts exemples de programes i molt proper al programador.
- **Accessibilitat:** el sistema operatiu oficial de la Raspberry Pi, Raspbian, inclou Python instal·lat i afavoreix la utilització d'aquest llenguatge de programació.
- **Intuïtiu i senzill:** Python disposa d'una sintaxis que afavoreix que el codi sigui senzill i fàcilment interpretat.

Python, a més, és un llenguatge de programació multi-paradigma, és a dir, permet al programador adoptar diferents estils de programació:

- **Programació orientada a objectes**: basada en encapsular dades i mètodes en estructures anomenades objectes, adquirint així un nivell més alt d'abstracció.
- **Programació imperativa:** és l'estil de programació que s'utilitza en l'assemblador, un nivell més proper a l'arquitectura del computador.
- **Programació funcional:** és un estil basat en la utilització de funcions i procediments que s'executen segons canvis d'estats governats per variables.

Una de les seves peculiaritats i el qual el diferencia d'altres llenguatges de programació és que és dinàmicament tipat, és a dir, una mateixa variable pot prendre valors de diferent tipus en diferents moments, fet que facilita en gran mesura la tasca del programador. Per altre banda, l'administració de

la memòria és automàtica i es realitza a través de la tècnica de recompte de referències, per tal de comptabilitzar les vegades que un determinat recurs ha estat referit i eliminar-lo quan ja no es fa servir ("Garbage Collector").

Una altre característica important de Python és que el sagnat, a diferència d'altres llenguatges com Java o C , és obligatori i indispensable, ja que s'utilitza per delimitar els diferents blocs i si no es realitza, el programa no s'executa i es marca com un error. Per altre banda, Python permet escriure nous mòduls fàcilment amb altres llenguatges de programació com és C o C++.

6.1.2. Instal·lació de les llibreries de Python en la Raspberry Pi

Per la instal·lació de les llibreries necessàries de Python en la Raspberry Pi, és necessari prèviament connectar-la a Internet. Una possible opció és connectar-la mitjançant un cable Ethernet a un router i com de sèrie l'adreça IP de la Raspberry Pi està configurada de forma dinàmica, automàticament s'obté connexió a internet.

Algunes llibreries de Python venen incloses en la instal·lació del sistema operatiu Raspbian, i es poden instal·lar a partir del comandament "sudo apt-get Update". No obstant això, altres més específiques no venen incloses, com ara "XlsxWriter" per crear documents d'Excel o "math" per realitzar operacions matemàtiques, aquestes llibreries cal utilitzar el comandament "sudo pip install" i el nom de la llibreria per tal de instal·lar-les.

6.2. Evologics DMAC Emulator

Per tal de poder realitzar proves d'enviament de dades i imatges amb els mòdems acústics de l'empresa EvoLogics sense necessitat de tenir-los físicament i haver d'alimentar-los, la companyia disposa d'un servidor virtual amb diversos mòdems configurables. La finalitat principal és permetre realitzar proves sense els mòdems físics, així com poder extreure un major número de mostres.

Per accedir al servidor cal realitzar una instal·lació prèvia, ja que és necessari disposar del software "OpenVPN", el qual és una eina de connectivitat punt a punt amb encriptació i validació d'usuaris per interconnectar els mateixos remotament de forma segura. Al instal·lar el software, es crea un "TAP Device" (TAP-Windows Adapter V9), és a dir, un adaptador virtual de Ethernet, el qual simula la funcionalitat d'una connexió real Ethernet entre dos dispositius (en aquest cas concret, l'ordinador per accedir als mòdems i el mòdem virtual en qüestió).

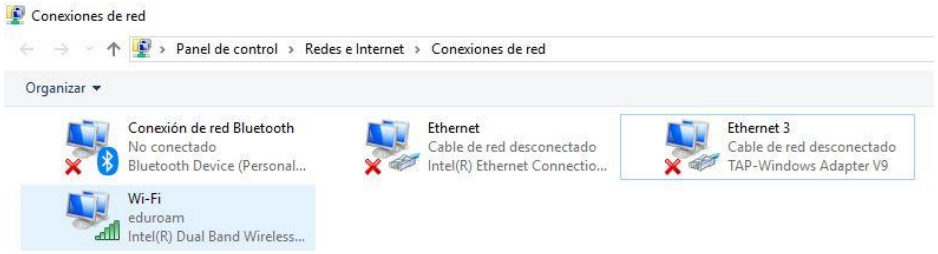

Figura 6.1. Adaptador virtual de Ethernet (TAP Device adaptar).

Tal i com es pot observar en la Figura 6.1 l'adaptador virtual "Ethernet3" està inhabilitat, ja que encara no s'ha establert la connexió amb el servidor virtual.

A partir de la instal·lació del software "OpenVPN" i executant l'arxiu de l'empresa EvoLogics que actua com enllaç entre l'usuari ("host") i el servidor virtual de la companyia (habilitació de l'accés a la xarxa virtual privada d'EvoLogics), es pot interactuar amb els mòdems virtuals mitjançant dos opcions: a partir d'un editor de text per programació (en aquest projecte consistirà principalment amb l'editor Spyder i la utilització d'un llenguatge de programació basat en Python), o mitjançant l'alternativa mencionada en el capítol 5 (veure capítol 5.2. Configuració Mòdems acústics EvoLogics S2CM 18/34) basada en l'eina de xarxa "NetCat" el qual permet, mitjançant l'intèrpret de comandes ("Shell") accedir als mòdems virtuals a través de les seves adreces IP i interactuar amb ells.

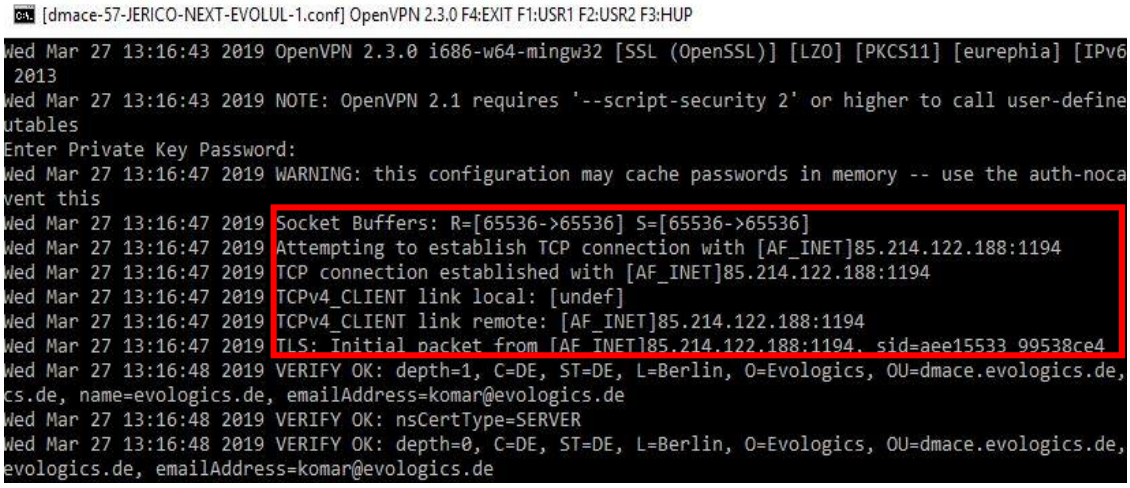

Figura 6.2 Representació del protocol de connexió TCP/IPv4 amb el servidor virtual (PC-servidor virtual EvoLogics DMACE Online)

En la figura 6.2 s'observa l'establiment de la connexió TCPv4 per tal d'accedir al servidor virtual d'EvoLogics.

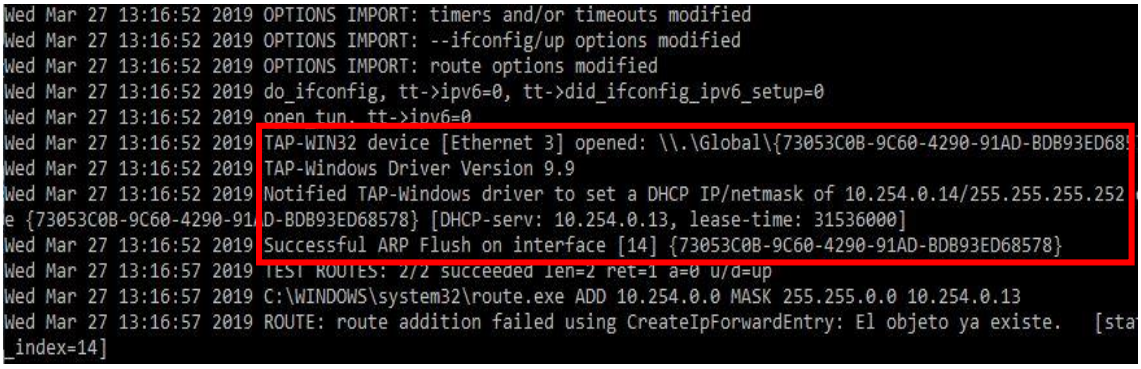

Figura 6.3 Habilitació de l'adaptador virtual Ethernet per l'establiment de la comunicació amb protocol TCP/IP amb el servidor virtual.

A partir de l'habilitació de l'adaptador virtual d'Ethernet (com es pot veure en la figura 6.3), s'estableix finalment amb èxit la comunicació amb el servidor virtual d'EvoLogics (DMACE Online) a través del protocol TCP/IP.

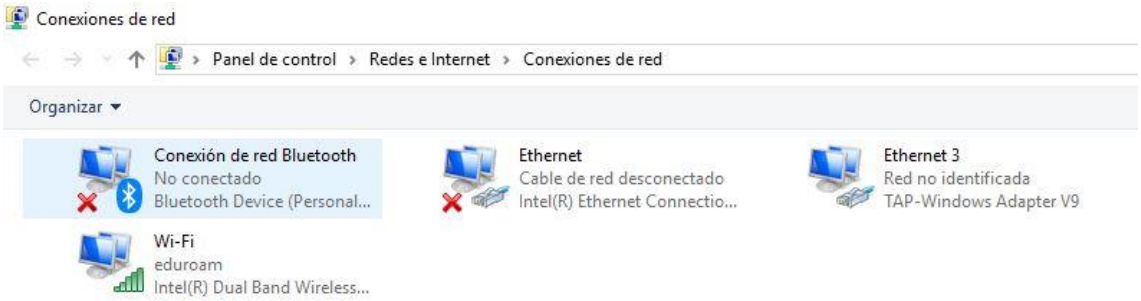

Figura 6.4 Habilitació de l'adaptador virtual Ethernet.

Una vegada queda habilitada la comunicació amb el servidor virtual d'EvoLogics, es pot accedir als mòdems virtuals a través de les seves direccions IP, així com els ports en els quals estan adreçades. Hi ha disponibles dos ports configurats, el 9200 i el 9201, i existeixen 10 possibles mòdems virtuals a utilitzar amb adreça IP: 10.42.57.n, amb n entre 1 i 10.

Cal mencionar que tot i ser una eina molt útil pel projecte, no es van poder fer les proves amb els mòdems virtuals a través de la Raspberry Pi pel retràs en la disponibilitat d'aquest dispositiu. Per tant, per tal de poder avançar amb el projecte es van dur a terme els programes mitjançant ordinadors fet que va suposar la utilització d'imatges d'internet i no de la pròpia càmera de la Raspberry. Aquest aspecte va ser probablement l'únic petit inconvenient en la utilització dels mòdems virtuals, ja que en el moment d'utilitzar els mòdems reals va ser necessari modificar certes línies del software corresponent a la captura de les imatges a través de la càmera de la Raspberry Pi, així com certs comandaments de Python que difereixen entre els sistemes operatius de Windows i Linux.

6.3. Software versió 1: Enviament i recepció d'imatges automàtic

Tal i com s'ha mencionat, l'objectiu d'aquest software és l'enviament automàtic d'imatges capturades per la càmera de la Raspberry Pi a l'equip receptor. Aquest software es divideix en dos: el programa encarregat de l'enviament de les imatges, que estarà executant la Raspberry Pi, i l'encarregat de la recepció de les mateixes, el qual estarà funcionant en l'ordinador del usuari que l'executi.

Una de les consideracions a tenir en compte abans de l'explicació del software, és que en cas de què el mòdem receptor no estigui alimentat, o el programa de recepció d'imatges no s'estigui executant, la Raspberry seguirà capturant imatges de forma automàtica i intentant enviar-les.

6.3.1. Programa d'enviament d'imatges automàtic

Aquest programa es pot dividir en quatre blocs principals:

- **Connexió i configuració del mòdem transmissor:** aquest bloc té la finalitat d'establir una connexió TCP/IP entre la Raspberry Pi i el mòdem acústic transmissor, així com de configurar el mòdem amb els paràmetres adients. Per establir la connexió entre la Raspberry Pi i el mòdem acústic és necessari obrir un "socket", que consisteix en l'enllaç entre un dispositiu i un altre mitjançant la definició de la IP del dispositiu, el port de comunicació i el protocol que utilitzen. Es fa servir principalment per comunicar-se amb altres programes que estan ubicats en diferents dispositius, com és per exemple en aquest projecte, la Raspberry Pi i el mòdem transmissor, o l'ordinador i el mòdem receptor.
- **Funcions d'enviament i recepció de dades:** per tal d'intercanviar informació entre el mòdem acústic transmissor i el receptor s'han definit funcions tant per l'enviament de dades, com per la recepció de les mateixes.
- **Captura i enviament d'imatges:** aquest bloc implementa tant la captura de les imatges com l'enviament de les mateixes.
- **Tractament de resultats i notificacions d'error:** un cop s'envien les imatges, per tal d'enregistrar els resultats, com són per exemple el temps mitjà d'enviament, la desviació estàndard o el percentatge d'imatges enviades correctament, es crea un fitxer Excel, així com un document de text per tal de recollir els possibles errors en l'enviament de les imatges.

En la figura 6.5 es mostra l'organigrama del funcionament del programa d'enviament d'imatges automàtic.

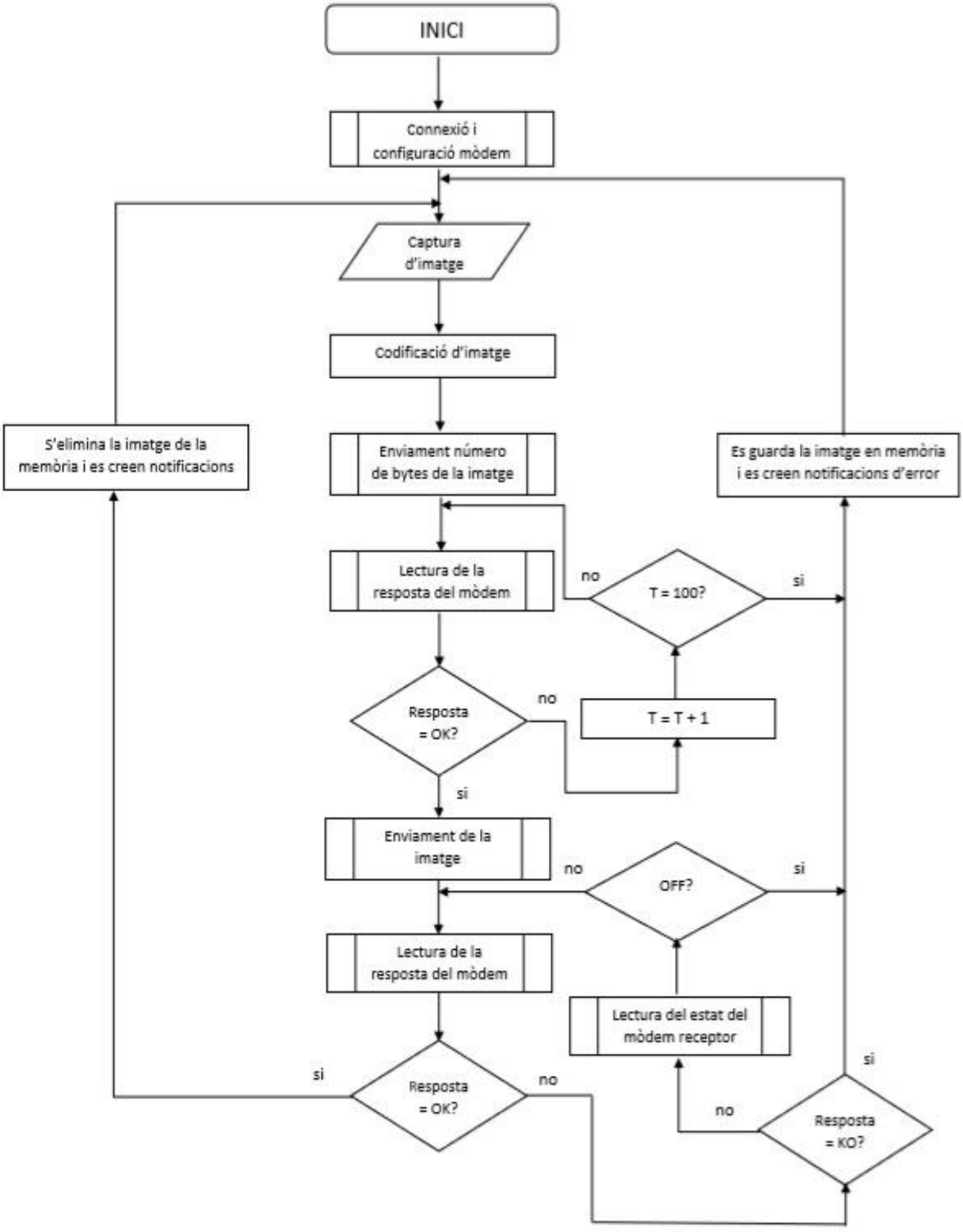

Figura 6.5 Organigrama del programa d'enviament d'imatges automàtic.

En primer lloc, s'estableix la connexió entre la Raspberry Pi i el mòdem acústic transmissor, per posteriorment realitzar la configuració del mòdem. A continuació, es captura la imatge a través de la càmera de la Raspberry Pi, i es codifica la imatge en base 64 per tal d'evitar problemes durant la transmissió, tal i com es mostra en la figura 6.6.

```
194     #Captació del temps de inici de captura de imatges
195     temps=time.time()
196     #Captura de la imatge
197     camera.capture(save_path1+'/'+str(temps)+'.jpg',quality=qualitat)
198     #Codificació de la imatge en base64
199     with open(save_path1+'/'+str(temps)+'.jpg',"rb") as imageFile:
200         imatge = base64.b64encode(imageFile.read())
```

Figura 6.6 Captura d'imatges i codificació.

El nom de la imatge s'estableix a partir de la marca de temps Unix (timestamp) generada per la funció time() i que permet determinar el moment en què ha estat capturada. El directori on es magatzemen també tenen per nom la data en què han estat capturades en format YYYY-MM-DD.

Posteriorment, s'envia el número de bytes de la imatge capturada, per així quan l'usuari rebi la imatge, detecti si s'ha rebut correctament o no. L'enviament del número de bytes va acompanyada d'un "flag" de símbol "#" per tal de què el mòdem receptor pugui distingir el número de bytes, de tota la informació que li arriba, així com d'altres dades irrellevants.

Per comprovar que el número de bytes ha estat rebut correctament, el mòdem receptor envia un missatge de "OK". Quan es detecta aquest missatge, es procedeix a enviar la imatge. En cas de no detectar el "OK" en un determinat període de temps, es detecta l'error de què el mòdem no està connectat i s'intenta capturar i enviar una altre imatge.

```
244     if t != 100:
245         i = 0
246         while i<len(nimatge):
247             if i == len(nimatge)-1:
248                 resposta3=modem2.send('AT*SEND,'+str(len(nimatge[i])+1)+
249                 ',3,'+str(nimatge[i]+"!"))
250             else:
251                 resposta3=modem2.send('AT*SEND,'+str(len(nimatge[i]))+
252                 ',3,'+str(nimatge[i]))
253             i = i+1
```

Figura 6.7 Enviament d'imatge.

Per l'enviament de la imatge, aquesta s'envia, dividida en paquets de 1024 bytes, al buffer del mòdem transmissor, (tal i com es va mencionar en el capítol anterior, veure capítol 5, apartat 5.3 Configuració Mòdems acústics EvoLogics S2CM 18/34) i per tal de senyalitzar el final de la imatge s'afegeix un flag amb símbol "!", tal i com mostra la figura 6.7.

```
100     #Funció per escriptura de comandes AT
101     def port_write(self,command):
102         while 1:
103             try:
104                 print 'Send',self.name +':',command
105                 self.netcat.sendall(command + '\n')
106                 break
107             except:
108                 try:
109                     self.netcat = socket.socket(socket.AF_INET,
110                     socket.SOCK_STREAM)
111                     self.netcat.settimeout(0.1)
112                     self.netcat.connect((self.ip, self.port))
113                     print "Connexio recuperada"
114                 except:
115                     print "Intent de connexio fallit"
116                     time.sleep(2)
117                     pass
118         return
```

Figura 6.8 Detecció de la desconnexió del enllaç entre la Raspberry Pi i el mòdem.

En la figura 6.8, es mostra la detecció de l'error de "socket", per tal de detectar la desconnexió de l'enllaç entre la Raspberry Pi i el mòdem acústic transmissor. Si es produeix un error en la comunicació Ethernet entre la Raspberry i el mòdem, el programa intentarà restablir la comunicació (tant en la recepció com en l'enviament de dades) en cas d'un microtall o d'una aturada momentània en la connexió, ja que aquest programa està pensat per executar-se de forma continuada en bucle, degut a què el sistema haurà d'estar submergit un temps indeterminat. Si aquesta desconnexió es produeix quan s'ha enviat la imatge i s'està esperant la resposta per part del mòdem receptor, el buffer del mòdem de transmissió es buida, i per tant, quan es recupera la connexió, el mòdem transmissor envia un flag d'alerta "$" per tal d'avisar al mòdem receptor que l'enviament ha estat aturat a causa d'un problema en l'alimentació, tal i com es pot veure en la figura 6.9.

```
302                     #Si es detecta que una pèrdua de la connexió, degut
303                     #a un microtall d'alimentació o de connexió ethernet
304                     #s'envia el flag "$" per informar al usuari que deixi
305                     #de llegir i esperi la nova recepció del numero de
306                     #bytes
307                     if x!=-1:
308                         aux1 =0
309                         modem2.send('AT*SEND,1,3,$')
310                         time.sleep(3)
311                         modem2.send("ATZ4")
```

Figura 6.9 Enviament del flag "$" per informar de la cancel·lació de la imatge.

Un cop la imatge ha estat enviada al buffer del mòdem transmissor, el programa espera la resposta de la correcta/incorrecta recepció de la imatge per part del mòdem receptor. Com les dades que constitueixen la imatge s'envien de cop al buffer del mòdem transmissor, i degut a què el temps d'enviament depèn tant de la mida de la imatge com de les condicions del medi, es comprova de forma simultània l'estat del canal acústic entre el mòdem transmissor i receptor, per tal d'així detectar una possible desconnexió per part del mòdem receptor i cancel·lar l'enviament, amb l'objectiu de minimitzar el temps d'espera.

Una altra possibilitat seria que el programa de recepció d'imatges es tanqués, però que el mòdem estigués alimentat i funcionant. Per tal de detectar aquesta situació, es comprova el número de bytes del buffer del mòdem transmissor: si aquest s'ha buidat, implica que la imatge ha estat enviada i, per tant, si en un cert període de temps no es rep cap resposta es determina la desconnexió per part de l'usuari.

```
297                 else:
298                     resposta = modem2.send("AT?S")
299                     aux1 =1
300                     x = resposta.find("INITIATION LISTEN")
301                     x2= resposta.find("1000000 bytes free")
```

Figura 6.10 Detecció de la desconnexió de l'enllaç entre el mòdem transmissor i el mòdem receptor.

En cas de no rebre cap resposta, es crea un Excel amb els errors i es guarda la imatge capturada en memòria. En cas contrari, si es rep el missatge "OK", implica que l'usuari ha rebut correctament la imatge, per tant s'elimina la imatge de la Raspberry Pi (per maximitzar l'espai) i es guarden els resultats de l'enviament de la imatge en l'Excel. Si es rep el missatge "KO", implica que l'usuari no ha rebut

correctament la imatge (el número de bytes esperats no coincideix amb els rebuts), per tant, es realitza el mateix procediment que en cas de no rebre resposta.

	A	B	C	D	E	F	G
1	Nom Imatge	Temps en (s)	Estat Imatge	Temps mitjà en (s)	Desviació estàndard	%correctes	Mostres
2	1557758746	62,45334	OK	60,95596	1,380733949	68,75	16
3	1557758751	60,39499	OK				
4	1557758757	63,44994	OK				
5	1557758763		OFF				
6	1557758768	61,49939	OK				
7	1557758774	60,49939	OK				
8	1557758780	60,32394	OK				
9	1557758785	61,29399	OK				
10	1557758791	62,19299	OK				
11	1557758797	59,03939	OK				
12	1557758802	60,32883	OK				
13	1557758808		KO				
14	1557758814		OFF				
15	1557758819	59,03939	OK				
16	1557758825		Disc.				
17	1557758831		OFF				
18							

Figura 6.11 Exemple d'Excel amb errors i resultats d'enviament d'imatges.

6.3.2. Programa de recepció d'imatges

Aquest programa es pot dividir en quatre blocs principals:

- **Connexió i configuració del mòdem receptor:** aquest bloc té la mateixa finalitat que el del mòdem transmissor, és a dir, establir una connexió TCP/IP entre l'ordinador i el mòdem acústic receptor, així com de configurar el mòdem amb els paràmetres adients.
- **Funcions d'enviament i recepció de dades:** per l'enviament i la recepció d'informació es defineixen les mateixes funcions que les utilitzades en el mòdem acústic transmissor.
- **Recepció i processament d'imatges:** aquest bloc fa referència a la recepció de les imatges transmeses pel mòdem acústic transmissor, així com el seu posterior processament per tal de recomposar la imatge amb les dades correctes.
- **Notificació de l'estat de recepció de les imatges:** per tal de notificar al mòdem transmissor la correcta o incorrecta recepció de les imatges, s'envien missatges d'estat.

A continuació, en la figura 6.12 es mostra l'organigrama amb l'esquema de funcionament del programa de recepció d'imatges automàtic.

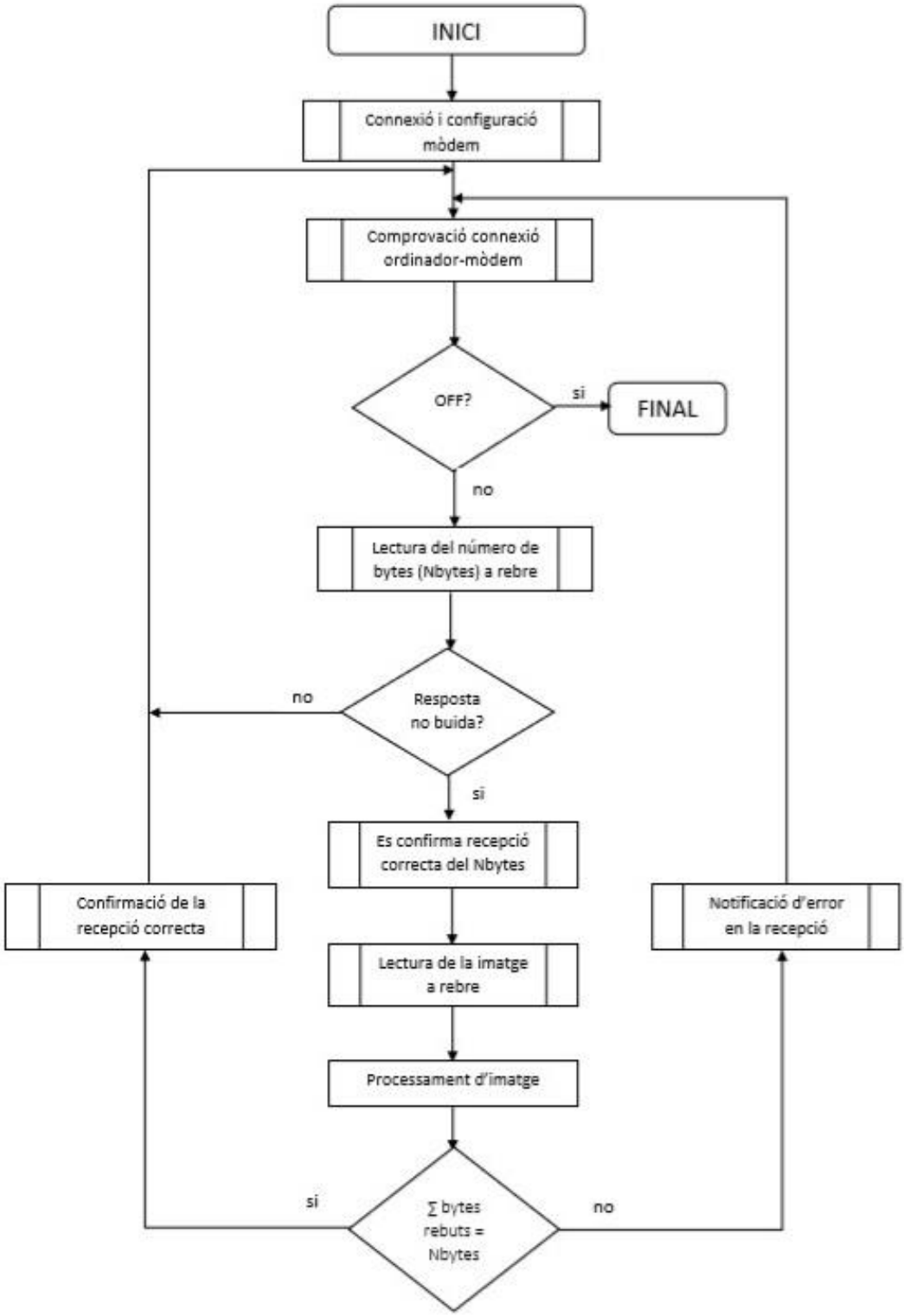

Figura 6.12 Organigrama del programa de recepció d'imatges automàtic.

Com en el programa d'enviament d'imatges, s'inicia amb la configuració del mòdem acústic receptor, així com l'establiment de la connexió entre l'ordinador i el mòdem, a través del socket. Posteriorment, es comprova l'estat de comunicació entre el mòdem receptor i l'ordinador; en cas que no estiguin comunicats, es tanca el programa, notificant-ho amb un missatge d'alerta, per tal d'evitar possibles errors en el funcionament del programa.

```
84              try:
85                  read = self.netcat.recv(1)
86              except socket.error as error:
87                  if str(error) != "timed out":
88                      print ("El programa es tancara degut a un" +
89                      "problema amb la alimentacio, comprovala i" +
90                      "executa de nou el programa")
91                      time.sleep(5)
92                      sys.exit(1)
93                      break
```

Figura 6.13 Avortament del programa en cas d'error en la comunicació entre el mòdem i l'ordinador.

Si la comunicació és correcta, es procedeix a llegir el número de bytes de la imatge a rebre, tenint en compte que aquest acaba amb el flag "#". Si no es rep cap resposta, es torna a comprovar l'estat de la comunicació i es torna a llegir el buffer del mòdem receptor.

Després de rebre el número de bytes, es notifica al mòdem transmissor la recepció correcta del número de bytes de la imatge, i es procedeix a acumular les dades corresponents a la imatge enviada pel mòdem transmissor, fins a rebre l'últim caràcter corresponent al flag "!" per detectar el final de la imatge, o fins a rebre el flag "$" corresponent a la cancel·lació de l'enviament a causa d'un problema en l'alimentació del mòdem transmissor.

Quan es detecta el flag "!" es procedeix a processar les dades obtingudes, eliminant tant aquelles que contenen informació irrellevant, com realitzant la descodificació de la imatge.

```
201         Imatgeaux = acc.splitlines()
202         mida= len(Imatgeaux)
203         aux2 = 1
204         i = 1
205         ImatgeNova = ""
206         #Es separen les diferents linies amb comes per el posterior
207         #tractament
208         while i<mida:
209             if aux2 == 1:
210                 ImatgeNova = ImatgeNova + Imatgeaux[0]
211                 aux2 = 0
212             ImatgeNova = ImatgeNova + ',' + Imatgeaux[i]
213             i=i+1
```

```
214         #Es realitza un altre split per comes i s'elimina la
215         #informació dels "DELIVERED" o "FALSE" els quals no aporten
216         #informació
217         ImatgeNova2 = ImatgeNova.split(",")
218         #Es busquen tots els "RECV" que precedeixen a la informació
219         #del missatge i s'eliminen (la capçalera)
220         m = [u for u, x in enumerate(ImatgeNova2) if x == "RECV"]
221         i=0
222         ImatgeDef = ""
223         while i<len(m):
224             #S'agafen 9 posiciones més ja que el missatge es troba 9
225             #posicions més endevant del "RECV"
226             ImatgeDef = ImatgeDef + ImatgeNova2[m[i]+9]
227             i=i+1
```

Figura 6.14. Processament de la imatge rebuda.

En la figura 6.14 es mostra el codi de programació pel processament de les dades que conformen la imatge enviada pel mòdem transmissor. Primerament es realitza una separació de les dades per salt de línies, i a continuació, es separen totes les dades obtingudes per comes. Posteriorment, es cerquen aquelles dades que contenen la capçalera "RECV", ja que aquestes contenen la informació de la imatge.

	Example RECV
AT?S	Requesting the acoustic connection status.
Local Address: 11	
Remote Address: 0	
Acoustic Link Status: INITIATION LISTEN	
Pool Status: 0 packages, 16384 bytes free	
IM Delivery Status: EMPTY	
Promiscuous Mode: ON	
RECV,4,11,8,8461,-40,120,65676,0.1000,test	The RECV notification.

Figura 6.15. Exemple de la informació obtinguda amb el "RECV". [12]

Com es pot observar en la figura 6.15, el missatge que s'envia amb la comanda "AT*SEND" (el que conté la informació de la imatge) es troba en la posició número 9 després del "RECV". Els altres paràmetres corresponen a informació referent a la longitud del missatge, l'adreça local, l'adreça de destí, velocitat de transmissió, RSSI, nivell d'integritat de la senyal, temps de propagació i l'indicador de qualitat del senyal. Per tant, tal i com es mostra en la figura 6.13, es realitza una concatenació de les dades corresponents a la posició on s'ubica el missatge, per tal d'obtenir les dades que conformen la imatge codificada.

Finalment, es descodifica la imatge i es comprova si la longitud de la imatge rebuda coincideix amb el número de bytes de la imatge esperada. En cas afirmatiu s'envia un missatge de "OK", en cas contrari un missatge de "KO".

6.4. Software versió 2: Enviament i recepció d'imatges configurable

Aquesta segona versió del software conté gran part dels blocs principals dels programes anteriorment descrits, ja que parteix de la mateixa estructura i base dels altres programes.

La finalitat d'aquest programa es controlar el número d'imatges que es volen capturar, així com la resolució de les mateixes i el grau de compressió de la imatge (qualitat). Un cop es configura un cert número d'imatges a enviar, la Raspberry comença a capturar les imatges i a intentar enviar-les a través del mòdem transmissor. Si el programa encarregat de rebre les imatges es tanca després d'haver realitzat la petició de les imatges, quan aquest es torna a executar el programa permet a l'usuari seguir rebent les imatges si el mòdem transmissor no ha acabat d'enviar-les. També es pot cancelar el procés si l'usuari ho desitja o realitzar una nova petició amb diferent configuració.

Per l'explicació del software d'aquesta versió s'ha prescindit de l'ús dels organigrames, ja que en ser un codi de programació més complex, aquests dificulten la seva comprensió. Per tant, a continuació es detallen les característiques més significatives que diferencien les dues versions, així com exemples per tal de facilitar l'explicació dels programes.

6.4.1. Programa d'enviament d'imatges configurable

Un cop s'ha realitzat la connexió TCP/IP entre la Raspberry Pi i el mòdem i aquest s'ha configurat, es crea el directori per emmagatzemar la primera sèrie d'enviament d'imatges, i el programa espera la recepció de la informació referent a la captura de les imatges.

```
205     while 1:
206         info = modem2.port_read()
207         x= info[0].find("!")
208         if x!=-1:
209             info = info[0].replace("!","")
210             break
211
212     #Creació de les series amb les dates
213     save_pathSerien=save_pathGlobalFotos+'/'+
214     str((str(NumeroSerie)).zfill(4))+'_'+str(datetime.date.today())
215     #Es guarda la informació obtinguda en diferents variables
216     info = info.split(",")
217     Nfotos =int(info[9])
218     Resolucio1= int(info[10])
219     Resolucio2= int(info[11])
220     Qualitat = int(info[12])
```

Figura 6.16 Obtenció de la informació relativa a les imatges a capturar.

De manera similar al cas de l'enviament de les imatges automàtic, es crea un directori amb un nom que identifica la sèrie i la data de captura i les imatges també s'anomenen amb un timestamp.

Un cop es rep la informació de la resolució, la qualitat i el nombre d'imatges a capturar, a partir del flag "!", es guarda la informació en 4 variables diferents. Nfotos, corresponent al nombre de fotos que es vol capturar, Resolucio1, corresponent al primer paràmetre de la resolució, Resolució2, corresponent al segon paràmetre de la resolució i finalment, Qualitat, referent al grau de compressió de la imatge JPG.

A continuació s'envien les Nfotos amb la qualitat i la resolució indicada, i es segueix el mateix procediment que l'explicat en el programa de la versió1, amb la diferència que mentre el mòdem transmissor envia la informació del buffer, es comprova, a més de la possible resposta del mòdem receptor així com la possible desconnexió, la recepció d'un flag "@" referent a la cancel·lació de les imatges (veure figura 6.17).

```
93              #Si es troba el flag de cancel·lació de l'enviament d'imatges
94              #"@" es retorna la variable aux per tal de interrompre
95              #l'enviament
96              x=read.find("@")
97              if x!=-1:
98                  aux=1
99              if read:
100                 buff += read
101             else:
102                 break
103         return buff,aux
```

Figura 6.17 Comprovació de la resposta del flag "@" corresponent a la cancel·lació de l'enviament d'imatges.

Finalment, un cop s'han enviat el número d'imatges de la sèrie corresponent, s'envia un flag "$" per notificar al mòdem receptor la finalització de l'enviament d'imatges. Aquest flag permet en cas de què el programa receptor es tanqui i el mòdem estigui en funcionament, detectar el final de l'enviament d'imatges de la sèrie en qüestió.

6.4.2. Programa de recepció d'imatges configurable

Com en la resta de programes, primerament es realitza la connexió TCP/IP entre l'ordinador i el mòdem, per posteriorment configurar-lo. A continuació, com es tracta d'un programa interactiu, és a dir, enfocat a l'usuari, es demana el número d'imatges a ser enviades, la resolució i la qualitat. En cas de no introduir valors correctes, es notifica i es torna a realitzar la petició d'introducció de les dades.

Es guarda en un document de text la informació de les fotos a enviar, així com el número de sèrie actual, per tal de què en cas de tancament del programa, es pugui detectar el número de sèrie, així com les fotos que s'havien configurat, amb l'objectiu de seguir rebent les imatges o cancel·lar-les (veure figura 6.18).

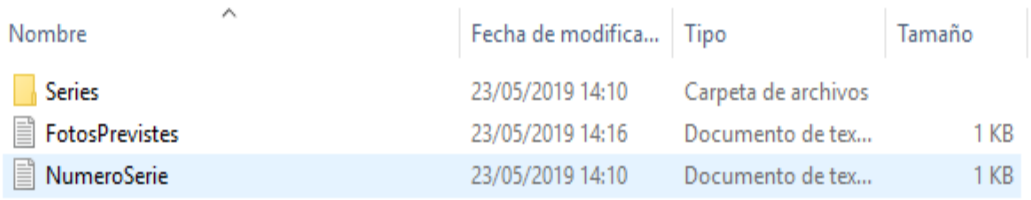

Figura 6.18 Número de sèrie, fotos previstes i conjunt de sèries amb les imatges rebudes.

A la figura 6.19 s'observa la comprovació de l'existència del fitxer corresponent amb la sèrie, per tal de detectar si ja hi havia configurat un enviament d'imatges previst o es tracta de la primera sèrie d'imatges.

```
307     respostax = os.path.exists(fitxerserie)
308     #En cas de no existir, es crea i es comença amb la primera serie
309     if respostax == False:
310         fh = open(fitxerserie,"w")
311         fh.write("1")
312         fh.close()
313         nserie = 1
314     #Si existeix, s'obre es llegeix i es continua amb la serie
315     #indicada
316     else:
317         fh = open(fitxerserie,"r")
318         nseriex = fh.read()
319         fh.close()
320         nserie = int(nseriex)
321
322     #Si existien imatges que havien sigut demanades anteriorment, i a
323     #causa del tancament del programa no es van poder rebre totes, es
324     #dona l'opció de continuar amb la recepció de les mateixes, o
325     #cancelar-les. Sinò, es comença una nova serie
326
327     fitxerfotos = os.path.join(save_pathGlobal,"FotosPrevistes.txt")
328     resposta = os.path.exists(fitxerfotos)
```

Figura 6.19 Comprovació de l'existència de fotos previstes.

Si ja s'havia introduït les dades però el programa es va tancar quan encara el mòdem transmissor estava enviant les imatges, es notifica a l'usuari la possibilitat de cancel·lar-les o de seguir rebent les imatges restants. Per tal de cancel·lar-les s'envia un flag "@", tal i com s'ha mencionat en l'explicació del programa d'enviament d'imatges. A més, per tal d'assegurar la finalització de la sèrie, es comprova l'estat del mòdem transmissor, per així no entrar en un bucle infinit. Aquesta funcionalitat es pot observar en la figura 6.20.

```
500                     #Per tal d'assegurar la finalització de la
501                     #captura es comprova tant l'estat del mòdem,
502                     #com s'envia un flag "@" per tal de comunicar
503                     #al mòdem la finalització de la serie
504                     while 1:
505                         aux4=0
506                         respostax = modem3.send("AT?S")
507                         x1 = respostax[0].find("INITIATION"+
508                         "LISTEN")
509                         if x1!=-1:
510                             aux4=1
511                         if respostax[1]== 1:
512                             aux4 = 1
513                         print ("Es procedeix a comunicar al"+
514                         "modem transmissor la cancelacio del"+
515                         "enviament de les imatges")
516                         resposta = modem3.send('AT*SEND,1,2,@')
517                         time.sleep(5)
518                         if resposta[1] == 1:
519                             aux4=1
```

Figura 6.20 Cancel·lació de l'enviament d'imatges.

De la mateixa manera que en el programa de recepció d'imatges automàtic, en cas de no establir una correcta comunicació entre el mòdem i l'ordinador, el programa es tanca per evitar problemes de mal funcionament del programa.

A partir d'una funció basada en el programa de recepció d'imatges automàtic, s'obtenen les imatges i es notifica tant els possibles errors com desconnexions.

En cas que l'usuari hagués configurat un enviament d'imatges i posteriorment tanqués el programa, quan es torna a executar, si l'usuari decideix continuar rebent les imatges, primerament es comprova si el mòdem transmissor ja ha finalitzat amb les imatges, a partir d'observar l'estat del canal acústic (veure figura 6.21). Si aquest no està tancat, posteriorment s'espera a la recepció del número de bytes de la següent imatge o la recepció del flag "$" corresponent a la finalització de l'enviament d'imatges de la sèrie corresponent.

```
429                 #Si es volen seguir rebent:
430                 if Delete == "0":
431                     condicio = 0
432                     while 1:
433                         #Es comprova l'estat del mòdem, per veure si
434                         #l'altre mòdem ja ha finalitzat, ha enviat el
435                         #flag, pero el mòdem receptor estava
436                         #desconnectat, i per tant, no ha pogut rebre
437                         #el flag
438                         while 1:
439                             respostax = modem3.send("AT?S")
440                             #Si el modem no es troba online,
441                             #significa que el mòdem transmissor ja ha
442                             #acabat l'enviament de les imatges
443                             x1 = respostax[0].find("INITIATION" +
444                             "LISTEN")
445                             if respostax[1] == 1:
446                                 condicio2 = 1
447                                 break
448
449                             if x1!=-1:
450                                 z=z+1
451                                 time.sleep(2)
452                                 if z==10:
453                                     condicio2 = 1
454                                     break
455                             else:
456                                 break
```

Figura 6.21 Comprovació de l'estat del mòdem transmissor, per continuar amb l'enviament d'imatges.

7. Resultats d'enviament d'imatges experimentals

A partir dels programes anteriorment explicats, es realitzen proves d'enviament d'imatges, per tal d'analitzar el temps d'enviament, així com els diferents paràmetres que afecten a aquest.

A continuació, es mostren proves experimentals amb els mòdems virtuals, utilitzant el servidor DMAC d'EvoLogics, així com proves amb els mòdems físics en un aquari.

7.1. Mòdems virtuals

A partir dels mòdems virtuals que disposa l'empresa EvoLogics i amb els procediments anteriorment explicats, en l'apartat 6.2 EvoLogics DMAC Emulator, s'ha procedit a realitzar un conjunt de proves d'enviament d'imatges amb mostres de diferent mida (en unitats d'imatges), resolucions (en pixels) i mides d'imatge (en bytes), per tal de prèviament a les proves reals amb la Raspberry Pi i els mòdems físics, poder fer una estimació dels temps d'enviament, així com de la desviació estàndard i el percentatge d'imatges enviades correctament.

Taula 7.1.- Resultats d'enviament d'imatges experimentals amb mòdems virtuals.

Nº	Imatge	Mida mostra	Temps mitjà (s)	Desviació estàndard (s)	Percentatge imatges correctes	Resolució (píxels)	Mida (bytes)
1	Color	1000	65,480	4,022	95,30%	800x479	52542
2	Color	1000	15,586	0,913	96,40%	284x177	8008
3	Color	100	277,038	3,745	92,00%	960x640	234485
4	Color	100	78,493	1,955	96,00%	640x480	63193
5	B&N	1000	11,713	0,941	99,60%	225x225	4184
6	B&N	100	363,392	3,849	91,00%	1174x810	348152
7	B&N	1000	142,364	2,039	94,54%	640x480	118014

Tal i com es pot observar en la taula 7.1 la correlació més significativa és la mida de la imatge (bytes) o pràcticament i directament proporcional la resolució (píxels) amb respecte el temps mitjà d'enviament. La relació entre ambdues característiques és pràcticament lineal, tal i com es pot observar en la figura 7.1.

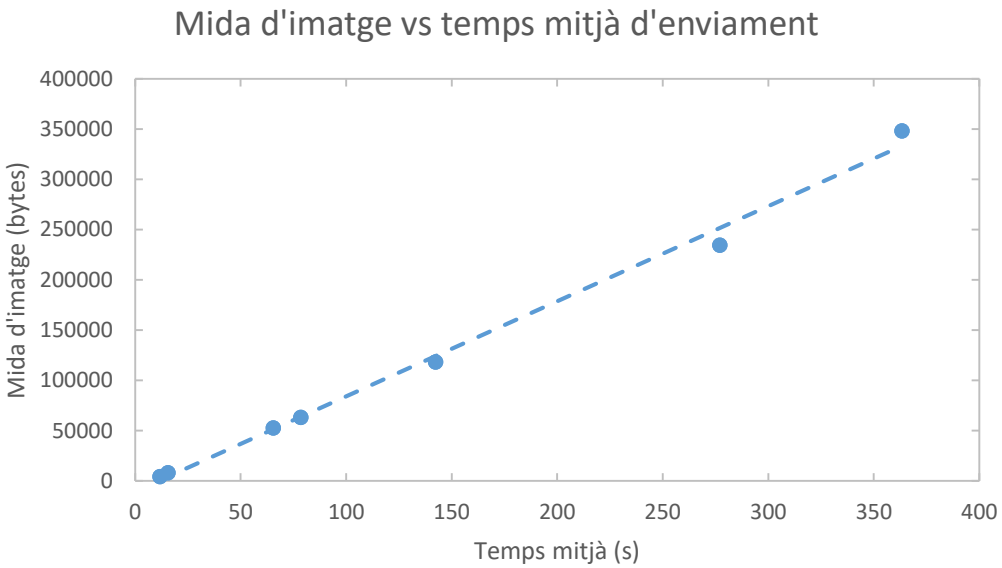

Figura 7.1.- Correlació amb tendència lineal de la mida d'imatge (bytes) i el temps mitjà d'enviament (segons) amb els mòdems virtuals.

Una altra conclusió que es pot extreure analitzant els resultats obtinguts en la taula 7.1 és la relació entre el percentatge d'imatges enviades correctament i la mida de la imatge. Com era d'esperar, amb major mida, el percentatge disminueix tal i com es pot observar en el gràfic 7.2, principalment degut a la pèrdua de paquets de bytes en l'enviament de les imatges. Per altra banda, el nombre d'imatges enviades no afecta negativament al percentatge d'imatges enviades correctament.

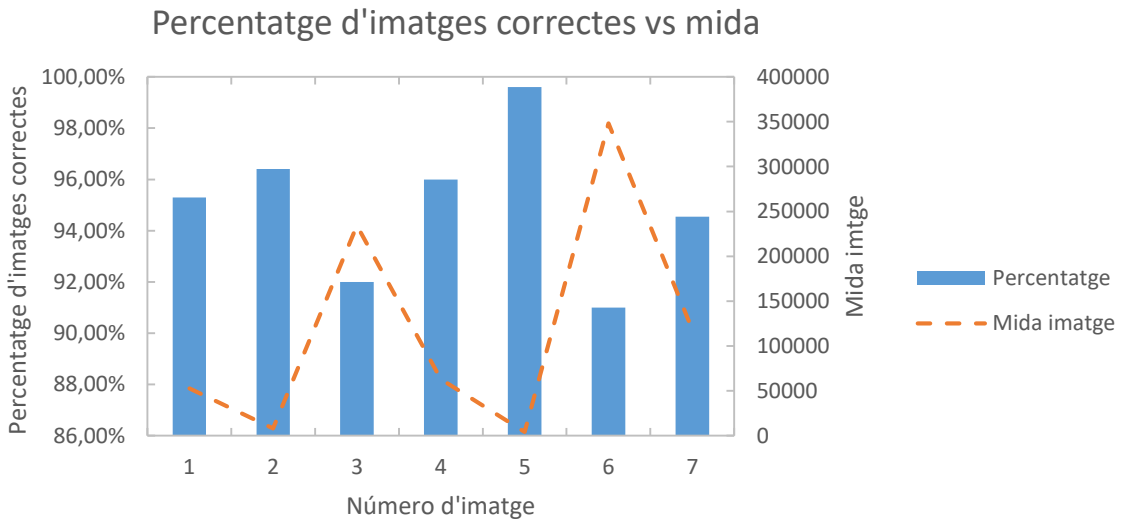

Figura 7.2.- Correlació entre el percentatge d'imatges correctament enviades i la seva mida en bytes amb els mòdems virtuals.

Pel que fa a la desviació estàndard, aquesta també manté relació amb el nombre de bytes de les imatges, ja que en gran part depèn principalment del nombre de paquets que fallen i es tornen a enviar (fet que fa augmentar aquest valor) per tant, amb major quantitat de paquets major serà la desviació estàndard associada al temps mitjà d'enviament d'imatges.

7.2. Mòdems físics

Per la realització de proves experimentals amb els mòdems físics, s'ha fet ús d'un aquari dins del recinte universitari així com proves sense immersió. Tot i així, per l'anàlisi de les dades i extreure conclusions no s'han pogut tenir en compte les proves realitzades fora del medi aquàtic, ja que aquests mòdems no estan dissenyats per utilitzar-los fora d'aquest entorn i amb proves de llarga duració es poden danyar i empitjorar el seu correcte funcionament. Per tant, en no poder realitzar suficients proves, aquestes no es consideren.

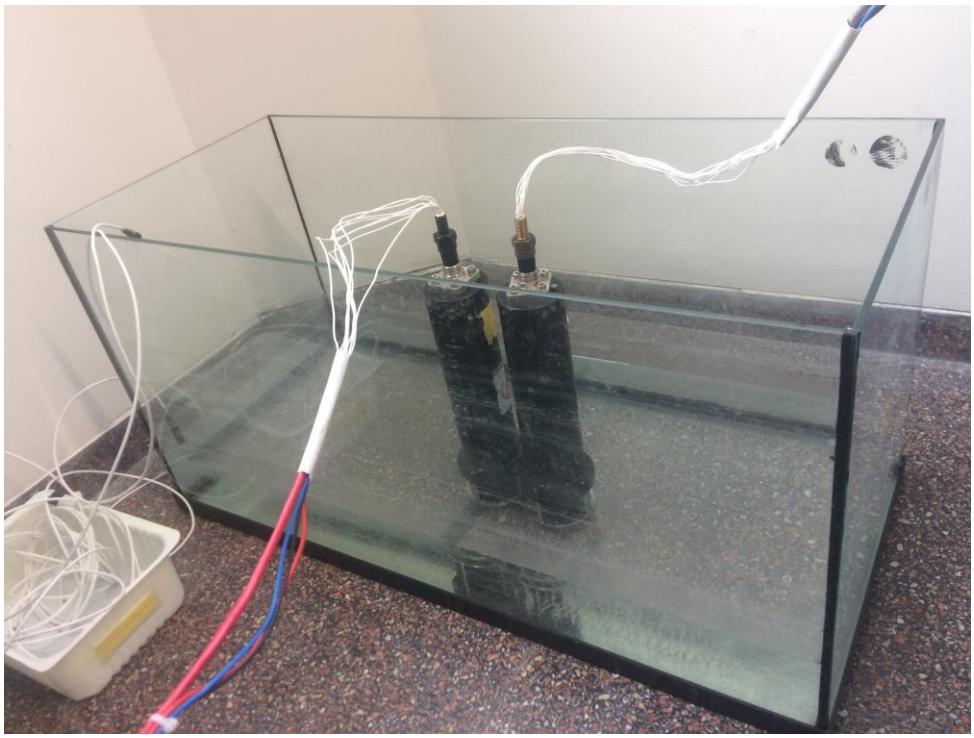

Figura 7.3.- Mòdems acústics físics en l'aquari per la transmissió i recepció d'imatges.

Les imatges capturades són de diferents resolucions i de diferents qualitats (grau de compressió del format JPG) per tal d'així obtenir resultats del temps d'enviament de les imatges amb respecte la mida.

A continuació, s'adjunten tres imatges capturades amb la càmera de la Raspberry Pi, amb mateixa resolució (640x480) però amb diferents qualitats (10%, 50% i 100%) per tal de mostrar l'efecte de la compressió de la imatge en JPG (com més alt el valor, menys grau de compressió).

Figura 7.4.- Imatge capturada amb la càmera de la Raspberry Pi amb resolució 640x480 i qualitat 10.

Figura 7.5.- Imatge capturada amb la càmera de la Raspberry Pi amb resolució 640x480 i qualitat 50.

Figura 7.6.- Imatge capturada amb la càmera de la Raspberry Pi amb resolució 640x480 i qualitat 100.

A partir de les imatges capturades, es realitza el mateix estudi que amb els mòdems virtuals, per tal d'analitzar tant el temps mitjà d'enviament, així com l'efecte de la mida de les imatges i la qualitat de les mateixes, en relació al percentatge d'imatges enviades correctament.

Taula 7.2.- Resultats d'enviament d'imatges experimentals amb mòdems físics.

Nº	Imatge	Mida mostra	Temps mitjà (s)	Desviació estàndard (s)	Percentatge imatges correctes	Resolució (píxels)	Qualitat	Mida (bytes)
1	Color	10	753,484	180,454	70,00%	640x480	10%	44539
2	Color	5	2275,495	390,897	40,00%	640x480	50%	135550
3	Color	10	1075,354	243,237	60,00%	720x480	10%	70349
4	Color	5	2534,485	421,953	30,00%	720x480	50%	165494
5	Color	10	985,451	196,485	50,00%	1280x720	10%	69458
6	Color	5	5750,483	500,404	20,00%	1280x720	50%	368450
7	Color	2	7833,384	100,348	0,00%	3280x2464	10%	479594

Com es pot observar en la taula 7.2, els temps d'enviament de les imatges són molt més elevats que els valors obtinguts amb els mòdems virtuals. Aquest fet és degut a les parets de l'aquari, les quals provoquen el fenòmen descrit en el capítol 3. Marc teòric, la multipropagació d'ones degut als rebots.

Aquests rebots generen multitud de camins pels quals viatgen els senyals acústics, provocant que el mòdem receptor no rebi de forma adequada les dades que el mòdem transmissor envia. Per tant, els temps d'enviament augmenten ja que el mòdem transmissor ha d'enviar, un cert número indeterminat de vegades, els mateixos paquets de 1024 bytes que conformen la imatge.

La relació entre el temps d'enviament i el número de bytes de la imatge segueix experimentant una relació lineal degut a què totes les imatges experimenten el mateix fenòmen de multipropagació d'ones pels rebots.

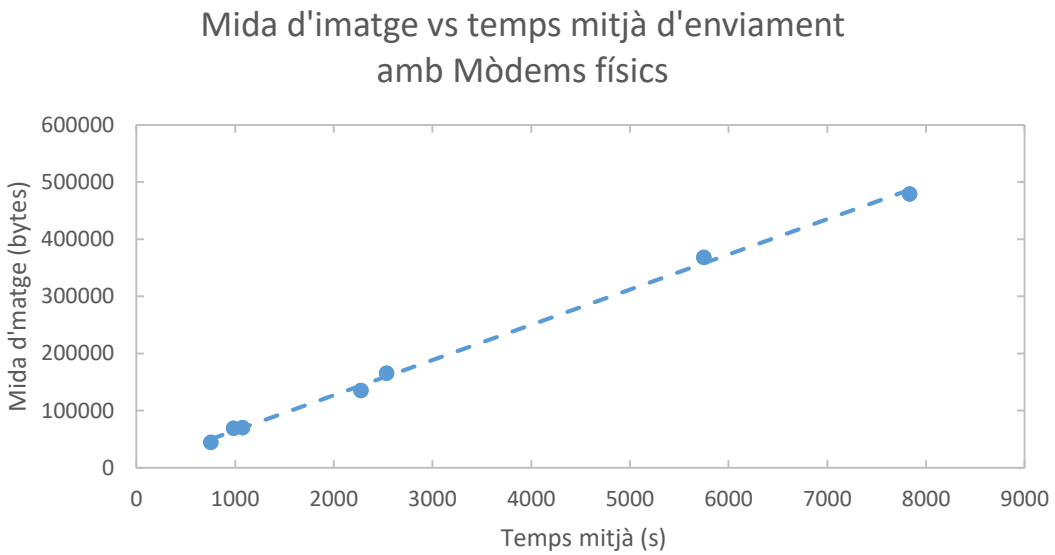

Figura 7.7.- Correlació amb tendència lineal de la mida d'imatge (bytes) i el temps mitjà d'enviament (segons) amb els mòdems físics.

Els percentatges d'imatges enviades correctament són molt inferiors en comparació als obtinguts amb els mòdems virtuals ja que, com s'ha mencionat, molts paquets de bytes fallen a causa dels rebots, i per tnat hi ha més probabilitat de què les imatges no s'enviïn de forma correcta. Tot i així, com s'observa en la figura 7.7, els percentatges de les imatges segueixen sent inferiors quan major és la imatge, degut a què hi ha més probabilitat de què més paquets de bytes fallin i per tant la imatge pugui no ser enviada correctament.

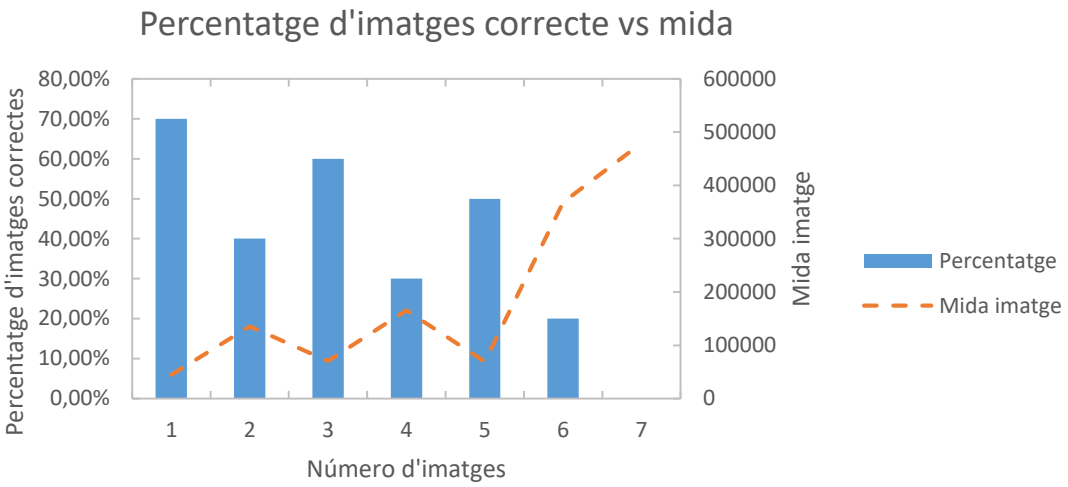

Figura 7.8.- Correlació entre el percentatge d'imatges correctament enviades i la seva mida en bytes amb els mòdems físics.

Per tal d'obtenir uns resultats d'enviament d'imatges correctes amb els mòdems físics, les proves s'haurien de realitzar a mar obert, per tal de minimitzar el fenòmen dels rebots, i així comprovar els temps reals respecte els temps d'enviament amb els mòdems virtuals. No obstant això, per manca de temps i degut a no disposar de tot el material del sistema dissenyat, aquestes proves no s'han pogut realitzar.

Cal destacar que tot i no haver realitzat suficients proves sense immersió amb els mòdems físics, els temps d'enviament de les imatges eren força inferiors als obtinguts en l'aquari, fet que confirma l'efecte dels rebots.

En el següent capítol, per tal d'estudiar la viabilitat del sistema, es tindran en compte els temps d'enviament de les imatges amb els mòdems virtuals, ja que aquests s'aproximen molt més als temps reals que els obtinguts amb l'aquari.

8. Consums energètics i viabilitat del sistema

Per tal d'extreure conclusions sobre la viabilitat del sistema dissenyat pelque fa al temps de funcionament, és necessari l'obtenció dels consums energètics del mòdem acústic receptor i de la Raspberry Pi connectada amb la càmera. Els consums energètics del mòdem transmissor, així com de l'ordinador, no es tenen en compte ja que aquests estaran directament alimentats per una font d'alimentació i no per les bateries.

En aquest capítol es mostrarà el circuit i el dispositiu utilitzat per tal de capturar les dades referents als consums energètics, així com el software i el programa per tal de mostrar aquestes dades i finalment valoracions sobre el prototip dissenyat i possibles millores d'aquest.

8.1. Adquisició dels consums energètics

Per tal d'obtenir els consums energètics del sistema, i en no disposar de les bateries anteriorment dissenyades ni dels reguladors de tensió, degut a la demora en l'entrega de les mateixes, els consums energètics s'han calculat a partir de fonts de tensió.

8.1.1. DAQ NI USB-6009

Per tal d'obtenir els consums energètics del sistema dissenyat s'ha decidit utilitzar un DAQ, és a dir, un dispositiu d'adquisició de dades, en concret el model de National Instruments, NI UBS-6009, degut principalment a la seva versatilitat i la seva fàcil i ràpida implementació. A més, per la resolució que es necessita aquest dispositiu ja és més que suficient.

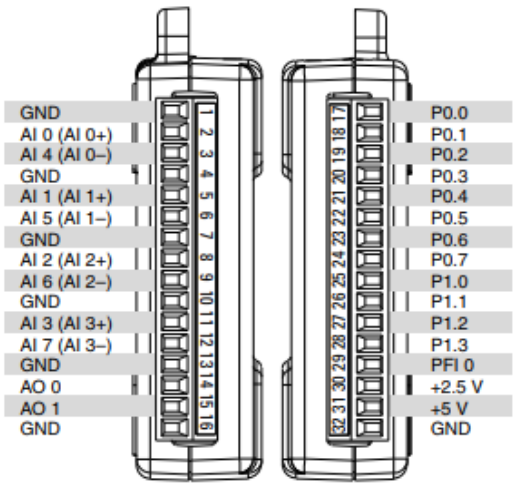

Figura 8.1. Pin-out del DAQ NI USB-6009 de National Instruments. [14]

Tal i com es mostra a la figura 8.1, el NI USB-6009, disposa de quatre entrades analògiques diferencials, 2 sortides analògiques i 12 entrades/sortides digitals. La limitació de voltatge de les entrades analògiques diferencials és de ±20 V i la limitació de tensió en respecte a GND és de ±10 V.

8.1.2. Consums energètics: Mòdem acústic

Per tal de capturar el valor de la intensitat que circula pel mòdem acústic és necessari la utilització d'una resistència de Shunt de valor de 1 Ω, ja que el DAQ tant sols mesura diferències de potencial, i per tant, al ser una resistència de 1 Ω la tensió és directament la intensitat.

Quan circula intensitat pel circuit hi ha una caiguda de tensió en la resistència de shunt provocant que part de la tensió de la font no sigui directament la tensió aplicada al mòdem acústic. La intensitat que circula pel mòdem pren dos valors: 0,05 A (en estat de recepció) i 0,1 A (en estat d'enviament). Aquest valors han estat mesurats directament amb un amperímetre. Aplicant una tensió de 24 V directament al circuit amb la resistència de shunt i el mòdem, la caiguda de tensió del mòdem variarà entre 23,95 V i 23,9 V. Al ser una diferència molt petita respecte els 24 V i donat que el límit del DAQ en tensions diferencials és de ±20 V, s'ha decidit aproximar el valor de la tensió a 24 V, ja que el consum energètic no es veurà pràcticament afectat. A més, mesurant directament la caiguda de tensió del mòdem transmissor sense la resistència de shunt, a partir d'un voltímetre, aquest valor es manté pràcticament constant a 24 V.

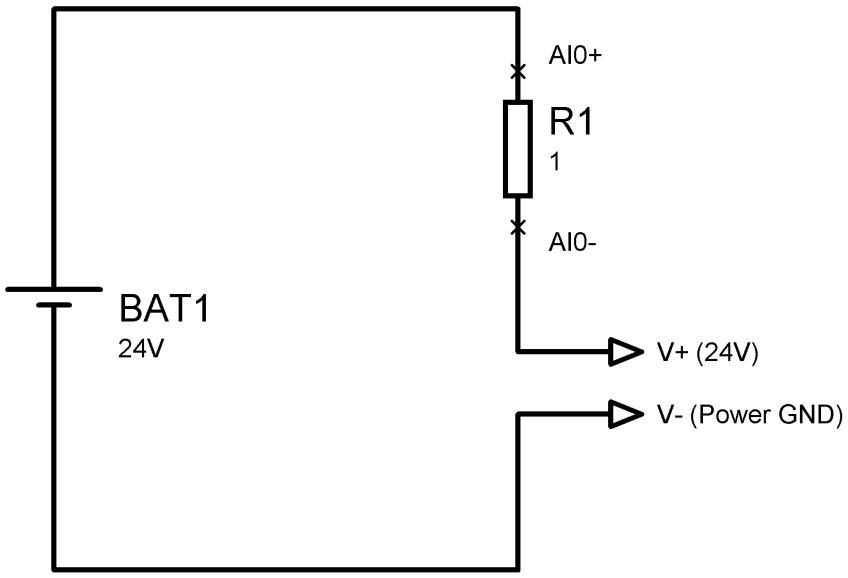

Figura 8.2. Circuit per mesurar el corrent del mòdem acústic amb el DAQ.

8.1.3. Consums energètics: Raspberry Pi amb càmera connectada

De la mateixa manera que en el cas del mòdem acústic, per tal de calcular la corrent que circula a través de la Raspberry Pi és necessari la utilització d'una resistència de shunt de 1 Ω. No obstant això, en aquest cas el corrent d'aquesta computadora de placa reduïda varia força entre 0,55 i 0,35 A degut a l'ús de la càmera per la captura de les imatges, així com en funció de l'estrès de la CPU pel programa que està executant.

En aquest cas, la caiguda de tensió que pateix la resistència de shunt és força més gran que en el cas anterior, fins a 0,5 V, alimentant la Raspberry Pi a 5 Vdc. Per tant, en aquest cas, la tensió que rep la Raspberry Pi és insuficient per tal de què funcioni. Per tant, cal realitzar un càlcul (tenint en compte la màxima tensió que pot suportar aquest dispositiu) de la tensió que és necessari aplicar amb la font d'alimentació per tal que la Raspberry Pi rebi una tensió suficient de 4,8 V i no excedeixi de 5,2 V.

Per 0,35 A, que és el consum mitjà de la Raspberry Pi (obtingut amb un amperímetre) sense l'ús de la càmera amb el programa executant-se, la caiguda de tensió a la resistència de shunt és de 0,35 V, per tant la tensió d'alimentació de la font hauria de ser d'aproximadament 5,35 V.

Amb una tensió d'alimentació de 5,4 V, la caiguda de tensió de la Raspberry Pi oscil·larà entre 4,9 V, en cas de circular la màxima intensitat, i 0,5 A i 5,1 V, en cas de circular la mínima intensitat de 0,3 A. Tot i que la captura de les gràfiques de tensió i corrent no seran les reals, degut a què la tensió hauria de ser constant a 5 V i la corrent sigués l'únic paràmetre que varis, la potència consumida com a resultat de la multiplicació de la tensió i la corrent, si serà un valor molt semblant al real, degut a què la tensió canvia proporcionalment amb la intensitat. Tot i així, aquests valors podrien diferir dels capturats amb un multímetre, degut a què són valors experimentals, amb valors resistius que tenen toleràncies, i no és el mètode més adient per calcular els consums energètics.

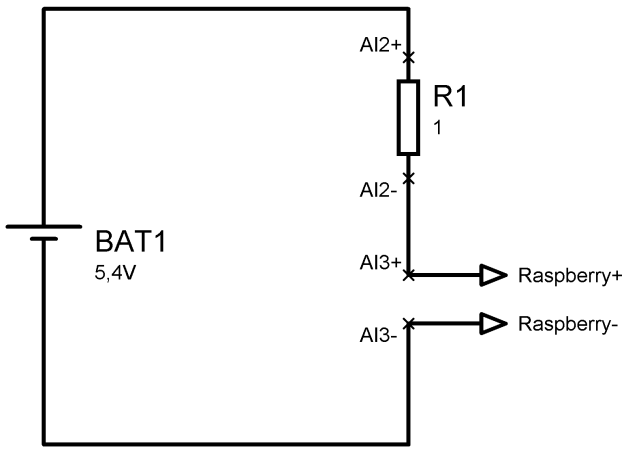

Figura 8.3. Circuit per mesurar tensió i corrent de la Raspberry Pi i la càmera amb el DAQ.

A la figura 8.4 es mostra el muntatge real del circuit amb el DAQ NI USB-6009, les resistències de shunt, les alimentacions amb la Raspberry Pi, la càmera, el mòdem i la connexió Ethernet.

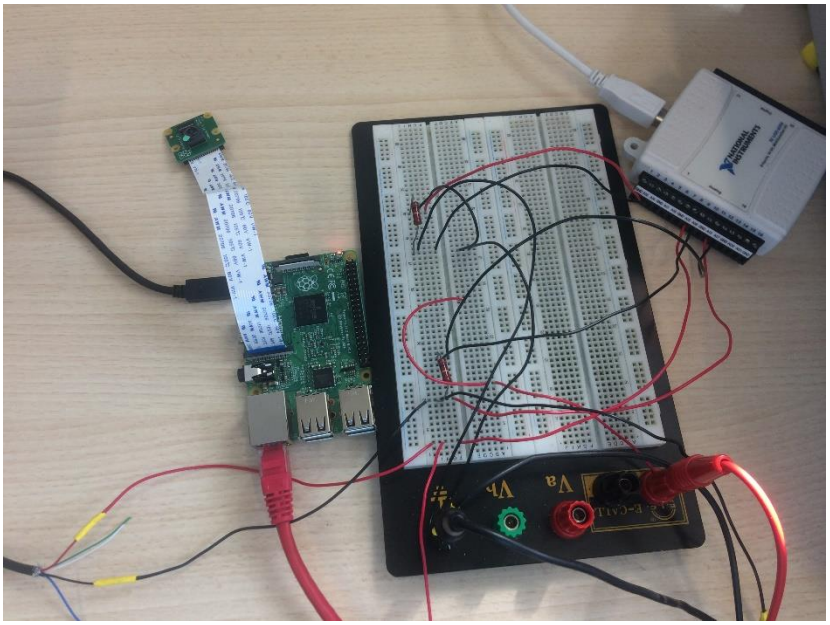

Figura 8.4. Circuit per mesurar tensió i corrent de la Raspberry Pi, la càmera i el mòdem amb el DAQ.

8.2. Presentació dels consums energètics amb LabVIEW

Per tal de mostrar els valors obtinguts a través del DAQ NI USB-6009, aquest es connecta mitjançant una connexió USB a un PC i mitjançant el software LabVIEW es realitza un programa per presentar aquestes dades.

8.2.1. Diagrama de blocs que constitueix el programa

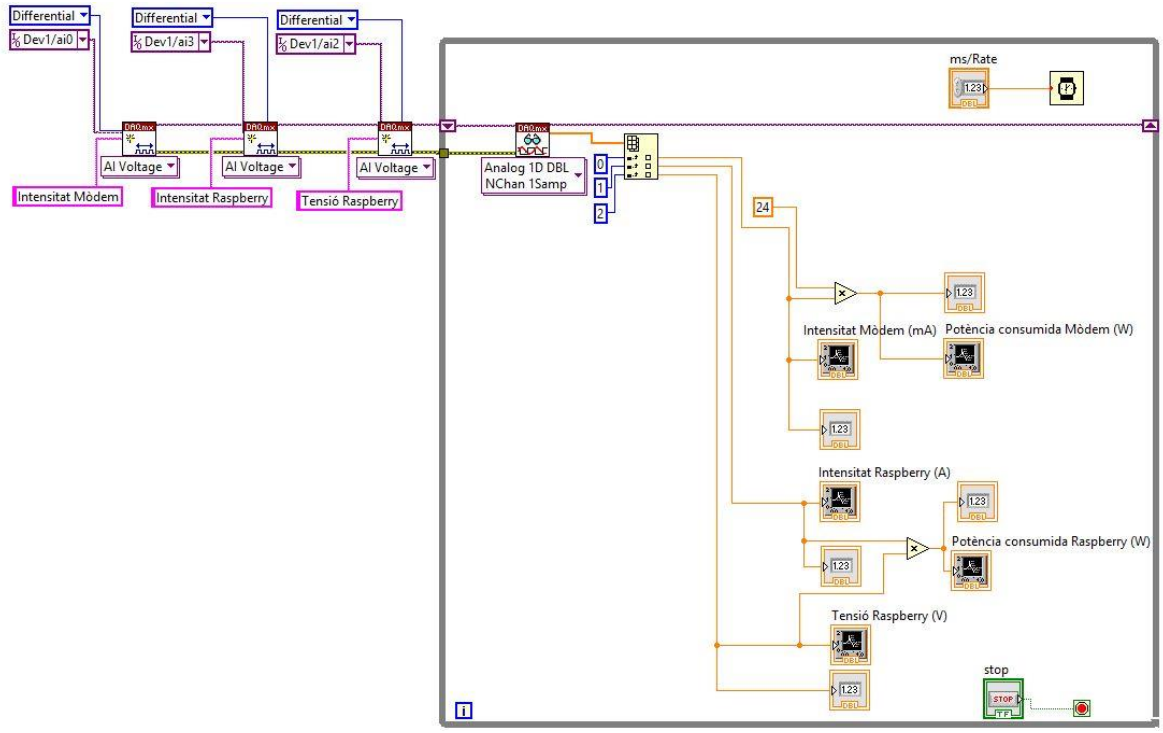

Figura 8.5. Programa en LabVIEW per mostrar els consums energètics capturats pel DAQ.

Tal i com s'aprecia a la figura 8.5, es mostra el diagrama de blocs que constitueix el programa per tal d'obtenir les dades del DAQ per a ser representades gràficament i numèricament. Es creen tres canals separats referents a la corrent del mòdem, la tensió de la Raspberry i la corrent. Posteriorment en un bucle while, es crea un bloc de lectura dels canals per tal de capturar simultàniament cada mesura. A continuació, mitjançant un "Index Array" es separen cadascun dels senyals per ser posteriorment tractats i mostrats amb gràfiques i indicadors numèrics.

Es crea també un botó Stop per finalitzar la captura de les dades, així com un "Numeric control" per tal d'establir el valor del temps en ms en què es prendrà cada mostra.

8.2.2. Interfície gràfica

En la figura 8.6, es mostra una interfície gràfica amb les cinc gràfiques corresponents a la tensió, la intensitat i la potència consumida de la Raspberry Pi amb la càmera connectada, així com la intensitat i la potència consumida del mòdem acústic transmissor. Aquest tipus de gràfiques tenen la peculiaritat que es poden exportar els valors a Excel i per tant realitzar els càlculs de potències mitges, així com realitzar gràfiques.

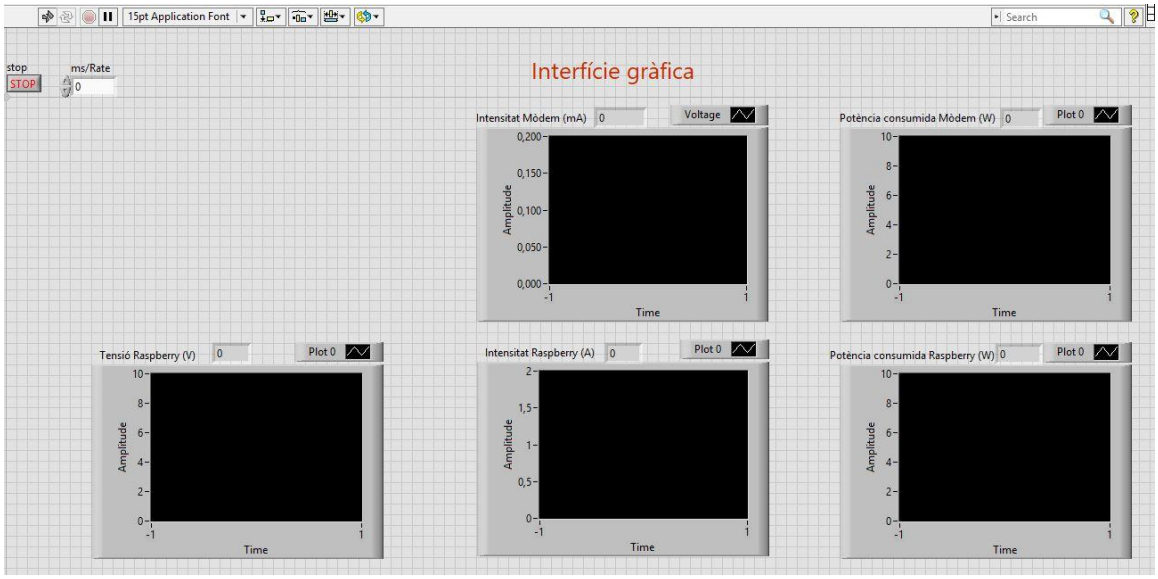

Figura 8.6. Interfície gràfica per la presentació dels consums energètics del sistema.

8.3. Resultats experimentals

A partir del programa anteriorment explicat i el sistema d'adquisició de dades, es realitza la captació dels consums energètics experimentals tant del mòdem acústic com de la Raspberry Pi, per tal d'extreure la potència mitja del mòdem acústic en repòs i en transmissió, així com el valor mig de la potència de la Raspberry Pi. La freqüència de mostreig del DAQ es configura en 10 mostres/s per tal d'obtenir suficients mostres en l'aproximació dels càlculs posteriors.

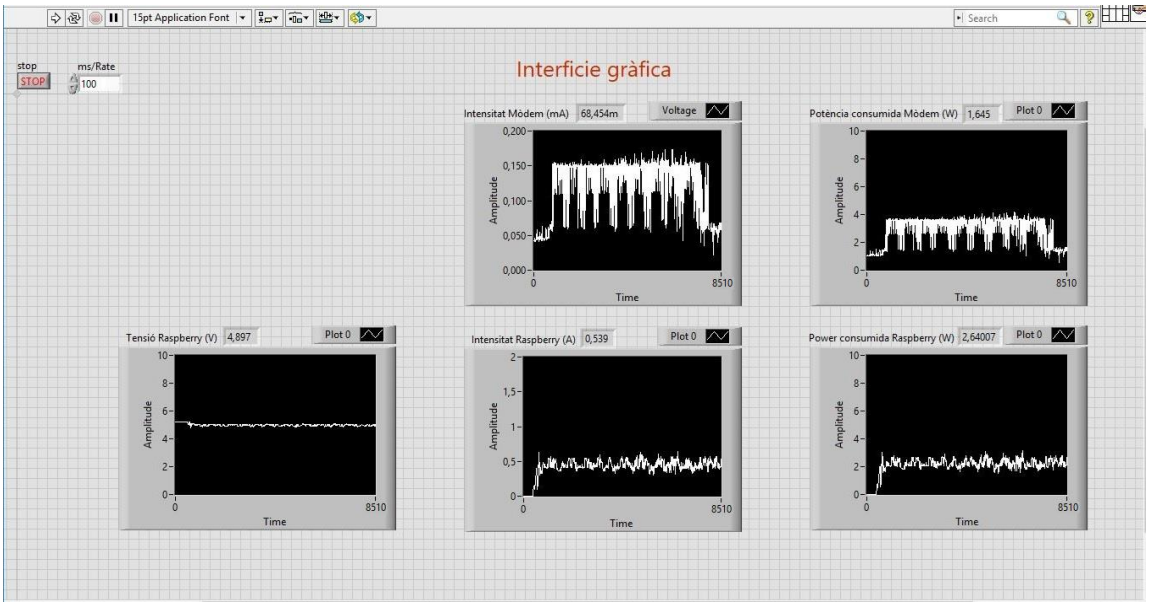

Figura 8.7. Resultats experimentals captats amb el programa de LabVIEW.

A partir dels valors obtinguts amb el DAQ, s'exporten les dades a Excel, i es realitzen les gràfiques, cambiant l'escala del temps en segons, per tal d'analitzar-les millor.

8.3.1. Consums energètics del mòdem acústic transmissor

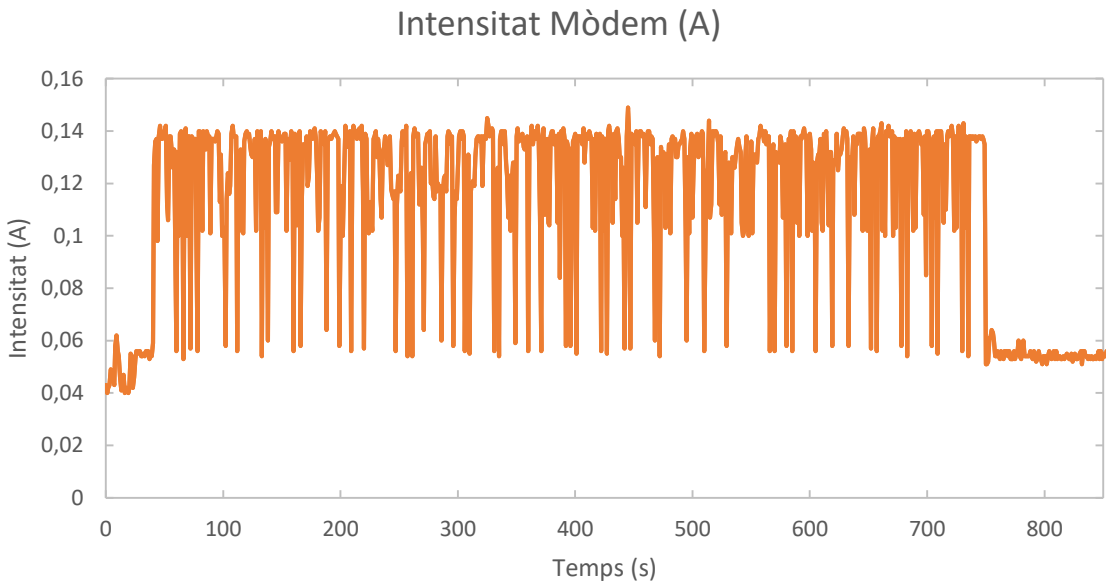

Figura 8.8. Intensitat en (A) del mòdem acústic transmissor.

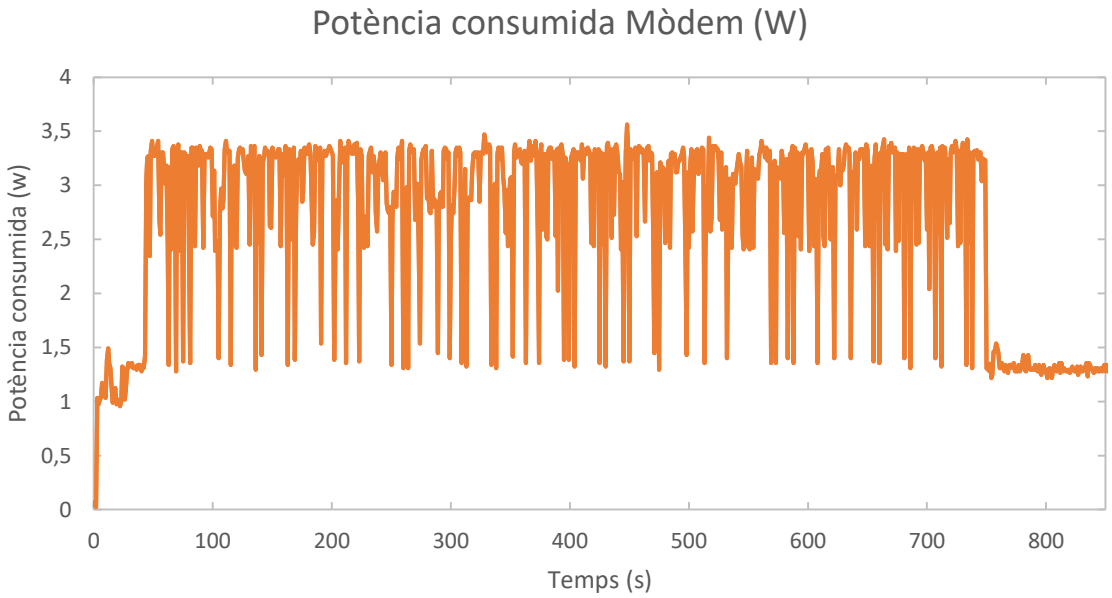

Figura 8.9. Potència consumida en W del mòdem acústic transmissor.

Pel que fa a la intensitat, com es pot observar en la figura 8.8, al principi fins el temps 50 s es mostra un valor d'aproximadament 0,054 A, fins que el mòdem comença amb la transmissió de la imatge. Com s'observa, a partir de què el mòdem comença la transmissió de la imatge, la intensitat pren dos valors constantment, un valor d'aproximadament 0,14 A i un de 0,054 A, degut a què entre paquet i paquet de 1024 bytes, el mòdem processa el següent paquet de bytes a enviar i durant aquell petit instant de temps consumeix com si estigués en mode de recepció d'informació.

En la figura 8.9 s'observa la potència consumida en W del mòdem acústic transmissor respecte el temps. Aquesta gràfica s'ha realitzat a partir dels valors d'intensitat capturats amb la resistència de shunt, multiplicats per una constant de valor 24, referent als 24 V de tensió del mòdem acústic.

Per tal d'obtenir uns valors mitjans aproximats del consum energètic del mòdem en repòs i el mòdem en mode d'enviament d'informació, es realitza una mitja dels valors de potència entre els instants de temps entre 50-750 pel consum en transmissió i entre 750-850 pel consum en recepció.

Taula 8.1. Potència consumida mitja del mòdem acústic transmissor.

Potència consumida mitja en recepció (Receive Mode)	**Potència consumida mitja en transmissió (Transmit Mode)**
1,285 W	2,642 W

Aquests consums energètics experimentals, es poden comparar amb els consums energètics establerts pel fabricant dels mòdems acústics, EvoLogics, el qual determina el consum per una distància inferior a 1000 m, de 1,3 W (o menys) en mode de recepció de dades, i 2,8 W (o menys) en transmissió de dades. Per tant, tot i ser uns valors obtinguts experimentalment amb components resistius no precisos i càlculs teòrics, aquests s'aproximen molt als valors reals.

8.3.2. Consums energètics de la Raspberry Pi i la càmera

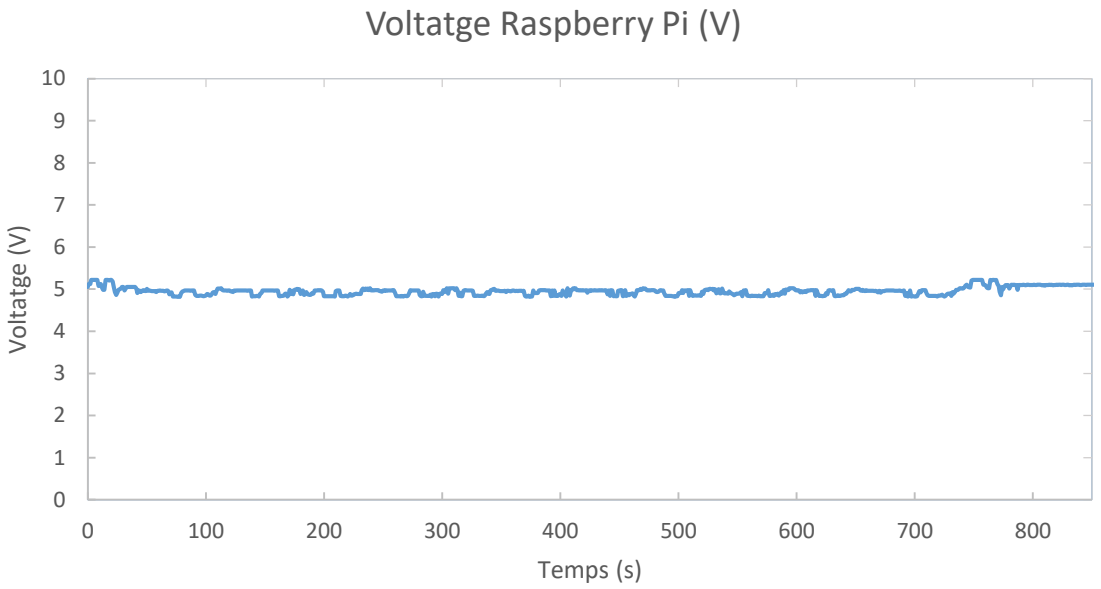

Figura 8.10. Caiguda de tensió en V de la Raspberry Pi i la càmera.

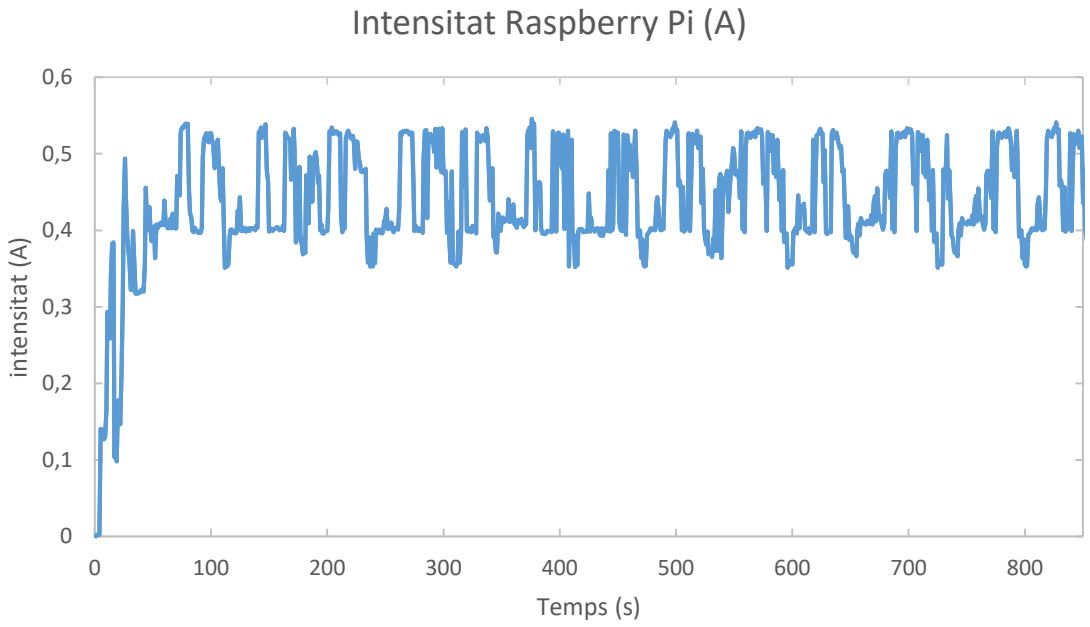

Figura 8.11. Intensitat en la Raspberry Pi i la càmera en A.

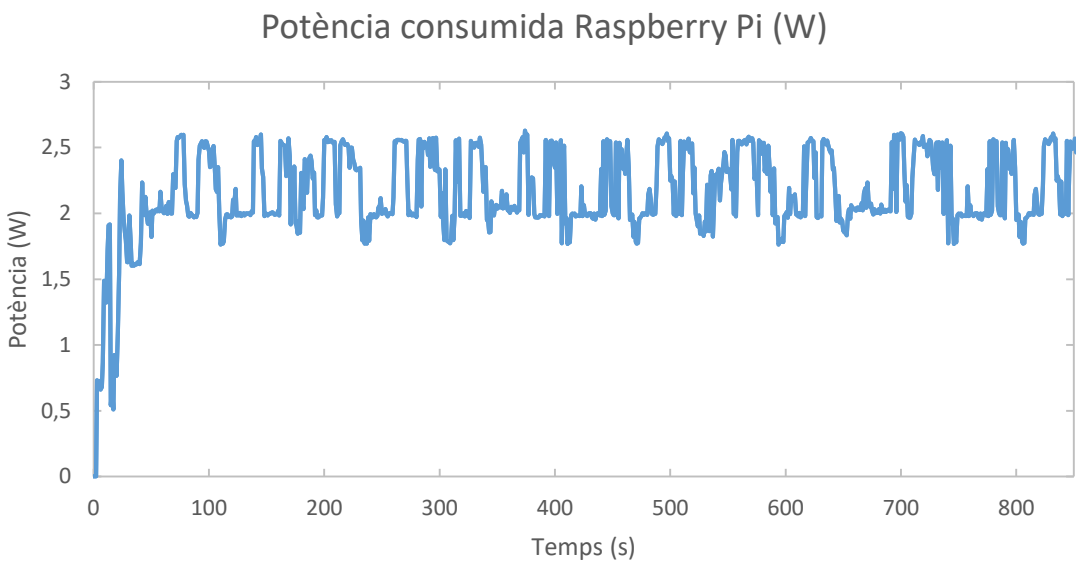

Figura 8.12. Potència consumida en W de la Raspberry Pi i la càmera.

Tal i com s'observa en la figura 8.10, la tensió oscil·la entre els valors calculats anteriorment, 4,9 V i 5,1 V, degut a la caiguda de tensió en la resistència de Shunt que varia respecte la intensitat. Per altra banda, la intensitat, tal i com s'aprecia en la figura 8.11, en el moment en què la Raspberry Pi es posa en funcionament, oscil·la entre 0,35 i 0,55 A, degut a què depèn de l'estrès de càrrega de la CPU, així com de la utilització de la càmera.

En la figura 8.12, es mostra la potència consumida en W de la Raspberry Pi. En aquest cas, la potència en transmissió d'imatges o sense és pràcticament constant, ja que en el moment en que el mòdem està en repòs, la Raspberry duu a terme certes operacions (tractament de la imatge, comprovació de paràmetres) que augmenten l'estrès de càrrega de la CPU. Per tant, en aquest cas es tindrà en compte tan sols un valor de consum energètic per la Raspberry Pi, tant quan el mòdem està en mode de transmissió com en recepció de dades.

Per calcular aquest valor, es realitza una mitja entre els instants de temps 70-850, ja que entre l'instant de temps 0-70 la Raspberry s'està iniciant. La potència mitja consumida de la Raspberry Pi amb la càmera s'ha determinat en un valor de 2,182 W.

La potència consumida de la Raspberry Pi experimental en comparació amb el valor teòric calculat a l'inici del projecte durant la selecció de les bateries (veure apartat 4.2.5 Bateries d'alimentació) és bastant inferior, degut principalment a què el valor teòric del consum de la càmera es va suposar com si aquesta s'utilitzés contínuament, ja que era l'únic valor que hi havia en la documentació tècnica. Els 2 W de la Raspberry Pi són correctes, no obstant això, la utilització de la càmera és puntual, i per tant no s'han de sumar directament els 1,25 W que seria si s'utilitzés de forma continuada.

8.3.3. Consum energètic total

En base als valors obtinguts en els subapartats anteriors, es conclou que la potència mitja consumida total es pot dividir en dos valors:

Taula 8.2. Potència total consumida mitja del sistema dissenyat en repòs i en transmissió.

Potència mitja consumida del sistema dissenyat en repòs.	Potència mitja consumida del sistema dissenyat en transmissió d'imatges.
3,467 W	4,824 W

Encara que aquests valors no són completament reals, per les aproximacions realitzades, s'aproximen força i permeten obtenir una perspectiva de la viabilitat del sistema dissenyat. A continuació i a partir d'aquests valors es realitzarà l'estudi de viabilitat sobre el temps de funcionament del sistema.

8.4. Estudi de viabilitat

A continuació es mostra un estudi de la viabilitat tant del prototip dissenyat com d'alternatives del mateix. Per a dur a terme aquest estudi s'ha considerat el temps el qual el sistema submarí pot estar en funcionament, com a variable per avaluar la seva viabilitat. Cal mencionar que s'han utilitzat els temps d'enviament de les imatges dels mòdems virtuals, degut a què aquests s'aproximen molt més als reals que els calculats amb els mòdems físics, a causa dels rebots provocats per les parets de l'aquari, tal i com s'ha mencionat anteriorment.

8.4.1. Sistema dissenyat

Per realitzar els càlculs dels dies d'ús de funcionament que podrà estar el sistema submarí dissenyat, s'ha utilitzat la següent equació:

$$t\ (\text{dies}) = \frac{Capacitat\ bateries(Ah) \cdot Tensió\ bateries\ (V)}{\left(\frac{Temps\ Enviament\ (s)}{\frac{Temps}{foto}(s)} \cdot Consum\ Actiu\ (W) + \left(1 - \frac{Temps\ Enviament\ (s)}{\frac{Temps}{foto}(s)}\right) \cdot Consum\ Repos\ (W)\right) \cdot 24\ h/dia} \quad (2)$$

El consum energètic del sistema es calcula realitzant la relació entre el temps d'enviament de la imatge i el temps que l'usuari vol enviar cada fotografia, respecte el consum actiu i el consum en repòs. Cal recordar que la capacitat de les bateries és de 18 Ah.

Taula 8.3. Nº de dies en funcionament del sistema vs freqüència de captura, resolució i qualitat de la imatge (10).

640x480 (10%)	50.000 bytes	1280x720 (10%)	70.000 bytes	3280x2464(10%)	480.000 bytes
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	4,602	1/70s	-	1/70s	-
1/90s	4,909	1/90s	4,602	1/90s	-
1/3min	5,558	1/3min	5,356	1/3min	-
1/5min	5,868	1/5min	5,861	1/5min	-
1/6min	6,058	1/6min	5,966	1/6min	-
1/8min	6,104	1/8min	5,985	1/8min	4,602
1/10min	6,203	1/10min	6,129	1/10min	5,019
1/30min	6,308	1/30min	6,282	1/30min	5,776
1/55min	6,352	1/55min	6,337	1/55min	6,046
1/1h	6,356	1/1h	6,343	1/1h	6,184
1/10h	6,399	1/10h	6,398	1/10h	6,369
1/15h	6,401	1/15h	6,400	1/15h	6,381
1/dia	6,402	1/dia	6,401	1/dia	6,390
1/5dies	6,403	1/5dies	6,402	1/5dies	6,402
1/10dies	6,404	1/10dies	6,403	1/10dies	6,403
1/mes	6,404	1/mes	6,404	1/mes	6,404

Taula 8.4. Nº de dies en funcionament del sistema vs freqüència de captura, resolució i qualitat de la imatge (50).

640x480 (50%)	140.000 bytes	1280x720 (50%)	370.000 bytes	3280x2464 (50%)	3.130.000 bytes
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	-	1/70s	-	1/70s	-
1/90s	-	1/90s	-	1/90s	-
1/3min	4,602	1/3min	-	1/3min	-
1/5min	5,186	1/5min	-	1/5min	-
1/6min	5,403	1/6min	4,602	1/6min	-
1/8min	5,613	1/8min	5,094	1/8min	-
1/10min	5,731	1/10min	5,349	1/10min	-
1/30min	6,163	1/30min	5,915	1/30min	-
1/55min	6,270	1/55min	6,128	1/55min	4,602
1/1h	6,281	1/1h	6,284	1/1h	4,954
1/10h	6,392	1/10h	6,378	1/10h	6,182
1/15h	6,396	1/15h	6,387	1/15h	6,255
1/dia	6,399	1/dia	6,393	1/dia	6,357
1/5dies	6,401	1/5dies	6,402	1/5dies	6,385
1/10dies	6,403	1/10dies	6,403	1/10dies	6,394
1/mes	6,404	1/mes	6,404	1/mes	6,404

En les taules 8.3 i 8.4 es mostra la relació entre el temps en què es va capturant cada imatge i el número de dies en què el sistema estarà en funcionament, respecte la resolució de la imatge així com la qualitat de la mateixa. Aquest últim paràmetre, la qualitat, fa referència al grau de compressió de la imatge en JPG. El primer valor (amb "Temps d'ús" no nul) de la columna "Freqüència captura (Hz)" fa referència

al temps mínim que es necessita per l'enviament de la imatge, és a dir, per una imatge amb resolució de 3280x2464 i una qualitat del 10%, es necessiten fins a 500 segons per la captura i l'enviament de la imatge, per tant, és el valor mínim de temps entre captura i captura.

Com s'observa en les taules anteriors, el temps màxim de funcionament del sistema varia en funció de la quantitat de bytes a transmetre, degut principalment a què com major és la mida de la imatge major és el temps que s'està consumint amb el consum energètic màxim.

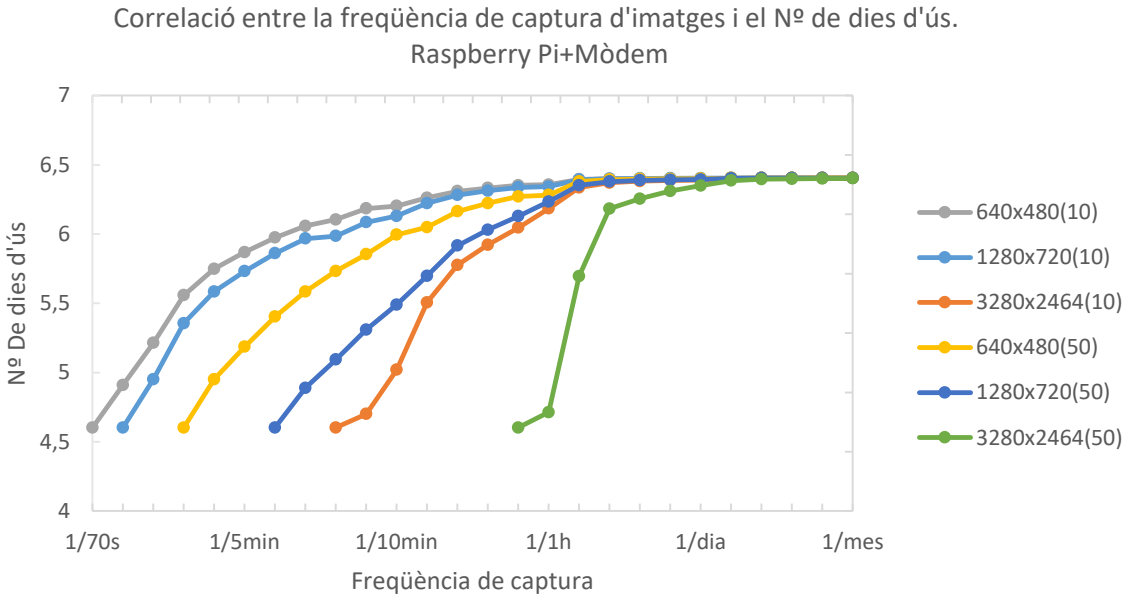

Figura 8.13. Correlació entre el número de dies d'ús i el número de fotos enviades del sistema dissenyat.

En la figura 8.13 es mostra la correlació entre el número de dies de funcionament del sistema submarí dissenyat amb respecte la freqüència de captura de les imatges, així com respecte la resolució de la imatge i la qualitat. Com s'ha mencionat anteriorment, quant major és la imatge (major resolució i major qualitat) menor número d'imatges es poden enviar per dia.

Com es pot apreciar, el temps de funcionament del sistema dissenyat quan s'envien imatges de forma seguida és de 4,602 dies. Aquest valor difereix força del calculat teòricament en l'apartat de la selecció de les bateries (veure apartat 4.2.5 Bateries d'alimentació), el qual era de 3,66 dies, degut principalment a què es va tenir en compte el consum de la càmera de forma errònia.

El número de dies de funcionament del sistema dissenyat pot ser vàlid en determinades aplicacions de curta durada, com ara en navegació. Tanmateix, en aplicacions com per exemple la mencionada en la motivació del projecte, el seguiment d'espècies, pot ser insuficient. Per tant la viabilitat del sistema és relativa en funció de l'objectiu de l'aplicació.

8.4.2. Alternatives del sistema dissenyat

Com ja s'ha mencionat, el temps podria ser insuficient per aplicacions que necessiten que el sistema estigui en funcionament durant llargs períodes de temps (superiors a 1-2 mesos). Per tant, en aquest subapartat es tractaran diferents alternatives per tal d'incrementar la viabilitat del sistema, així com optimitzar-lo.

- **1ra alternativa, Raspberry Pi + utilització d'un relé d'estat sòlid:** una alternativa vàlida seria la utilització d'un relé d'estat sòlid, per tal de controlar l'encesa i l'aturada del mòdem acústic transmissor, a través dels pins GPIO de la Raspberry Pi, en els moments en què aquest no s'utilitza. El relé hauria de ser d'estat sòlid per tal d'així aconseguir minimitzar el consum energètic. Aquesta alternativa es contempla ja que els mòdems acústics no disposen del mòdul "Wake-up" instal·lat, el qual realitzaria aquesta mateixa finalitat però controlat tot per programació.

  El mòdem es pot deixar d'alimentar a través del seu connector "Power ON (3)", si aquest no es connecta directament al negatiu de l'alimentació.

  Una possibilitat d'elecció del relé seria la utilització del model CPC1114N de l'empresa IXYS Corporation, el qual es tracta d'un relé d'estat sòlid OptoMOS normalment tancat.

Electrical Characteristics @ 25°C

Parameter	Conditions	Symbol	Min	Typ	Max	Units
Output Characteristics						
Load Current						
Continuous [1]	I_F=0mA	I_L	-	-	400	mA_{rms} / mA_{DC}
Peak	I_F=0mA, t≤10ms	I_{LPK}	-	-	1000	mA
On-Resistance [2]	I_F=0mA, I_L=400mA	R_{ON}	-	1.2	2	Ω
Off-State Leakage Current	I_F=2mA, V_L=60V_P	I_{LEAK}	-	-	1	µA
Switching Speeds						
Turn-On	I_F=5mA, V_L=10V	t_{on}	-	0.46	2	ms
Turn-Off		t_{off}	-	2.1	5	
Output Capacitance	I_F=2mA, V_L=50V, f=1MHz	C_{OUT}	-	24	-	pF
	I_F=2mA, V_L=1V, f=1MHz		-	114	-	
Input Characteristics						
Input Control Current to Activate (Output Open) [3]	-	I_F	-	0.58	2	mA
Input Control Current to Deactivate (Output Closed)	I_L=120mA	I_F	0.1	0.56	-	mA
Input Voltage Drop	I_F=5mA	V_F	0.9	1.2	1.5	V
Reverse Input Current	V_R=5V	I_R	-	-	10	µA
Common Characteristics						
Capacitance, Input to Output	V_{IO}=0V, f=1MHz	C_{IO}	-	1	-	pF

Figura 8.14. Especificacions elèctriques del relé d'estat sòlid NC CPC1114N. [15]

Absolute Maximum Ratings @ 25°C

Parameter	Ratings	Units
Blocking Voltage	60	V_P
Reverse Input Voltage	5	V
Input Control Current	50	mA
Peak (10ms)	1	A
Input Power Disipation [1]	70	mW
Total Power Dissipation [2]	400	mW
Isolation Voltage, Input to Output	1500	V_{rms}
ESD Rating, Human Body Model	8	kV
Operational Temperature	-40 to +85	°C
Storage Temperature	-40 to +125	°C

Figura 8.15. Especificacions elèctriques màximes del relé d'estat sòlid NC CPC1114N. [15]

Tal i com s'observa en les figures 8.14 i 8.15, el corrent de control que ha de circular a través del LED per tal de què els contactes del relé s'obrin ha de ser com a màxim de 50 mA. El fabricant estableix que a partir d'un corrent de control de 5 mA, la caiguda de tensió ha de ser d'un valor entre 0,9 i 1,5 V, sent el valor típic 1,2 V. També cal remarcar que el corrent màxim de la càrrega a la sortida s'estableix en 400 mA, valor més que suficient pel nostre disseny.

Per tant, configurant un dels pins GPIO (en concret el 17) com a sortida, els quals subministren 3,3 V amb un corrent màxim de 16 mA, és necessari col·locar una resistència en sèrie per tal de limitar el corrent a 5 mA i assegurar una caiguda de tensió en el LED de 1,2 V. Així doncs, la resistència es pot calcular com:

$$R = \frac{Vrasp - Vf}{IF} = \frac{3{,}3\ V - 1{,}2\ V}{5 \cdot 10^{-3}\ A} = 420\ \Omega \qquad (3)$$

Seleccionant un valor comercial de la resistència pròxim als 420 Ω es selecciona un resistor de la sèrie E96, amb tolerància d'un 1% de 422 Ω.

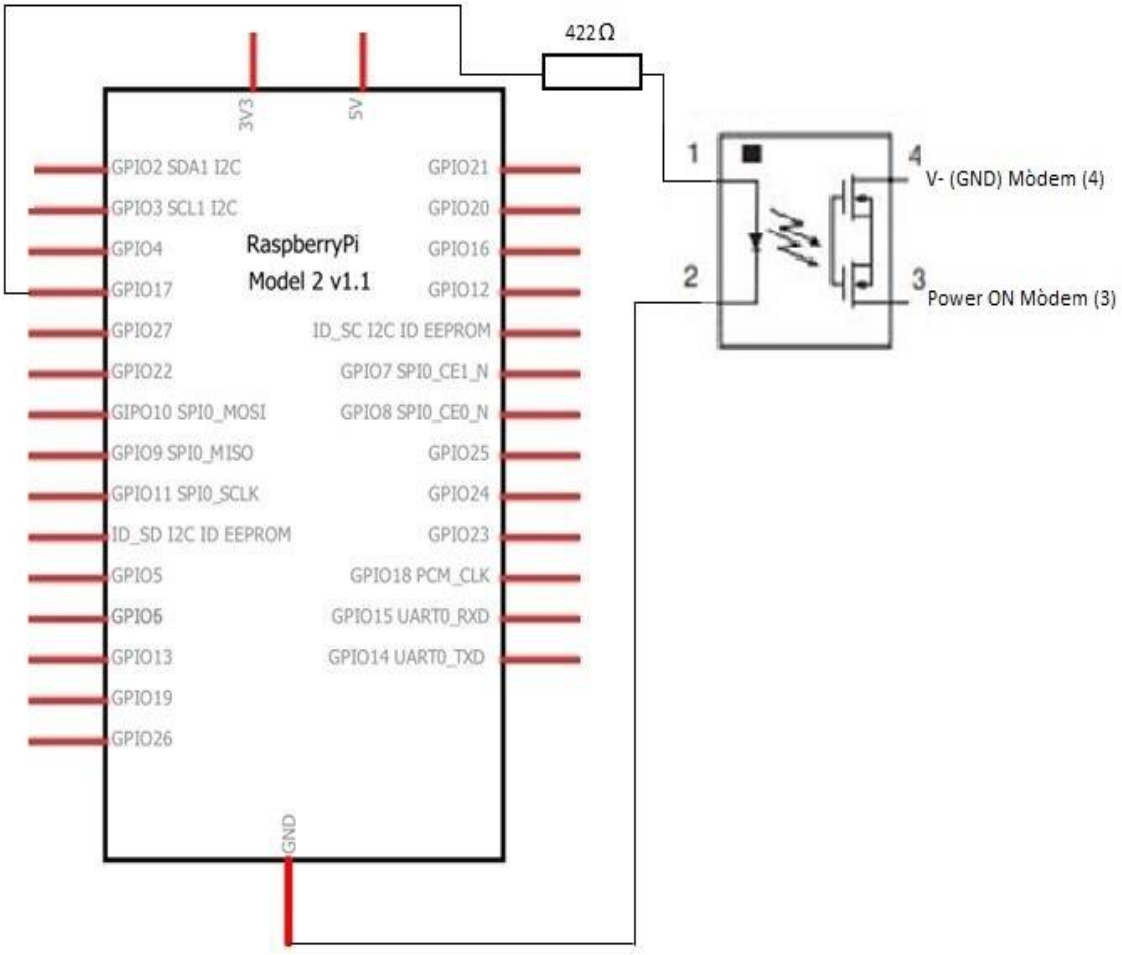

Figura 8.16. Esquema de connexió del pin GPIO de la Raspberry Pi i el relé d'estat sòlid.

La potència consumida pel relé d'estat sòlid seleccionat es pot calcular com:

$$P(W) = I \cdot V = 5 \cdot 10^{-3} A \cdot 1{,}2\ V = 0{,}006\ W \qquad (4)$$

Per tant, la potència que consumeix el relé d'estat sòlid, com era d'esperar és despreciable.

Mitjançant aquest relé, el mòdem transmissor i la Raspberry Pi, es pot dissenyar un sistema amb un consum energètic força inferior que el prototip original. Utilitzant el consum energètic del sistema en transmissió de dades de 4,284 W (igual que l'original) i el consum en repòs, de 2,182 W, de la Raspberry Pi, considerant nul el consum del mòdem mitjançant el relé, es poden realitzar els nous càlculs de la duració del funcionament del nou sistema:

Taula 8.5. Nº de dies en funcionament del sistema (alternativa 1) vs freqüència de captura, resolució i qualitat de la imatge (10).

640x480 (10%)	50.000	1280x720 (10%)	70.000	3280x2464 (10%)	480.000
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	4,602	1/70s	-	1/70s	-
1/90s	5,240	1/90s	4,602	1/90s	-
1/3min	6,918	1/3min	6,338	1/3min	-
1/5min	7,934	1/5min	7,464	1/5min	-
1/6min	8,321	1/6min	7,908	1/6min	-
1/8min	8,701	1/8min	8,355	1/8min	4,602
1/10min	8,916	1/10min	8,612	1/10min	6,294
1/30min	9,718	1/30min	9,595	1/30min	7,615
1/55min	9,922	1/55min	9,851	1/55min	8,560
1/1h	9,942	1/1h	9,877	1/1h	9,105
1/10h	10,152	1/10h	10,146	1/10h	10,008
1/15h	10,161	1/15h	10,156	1/15h	10,064
1/dia	10,166	1/dia	10,163	1/dia	10,105
1/5dies	10,174	1/5dies	10,174	1/5dies	10,162
1/10dies	10,175	1/10dies	10,175	1/10dies	10,169
1/mes	10,177	1/mes	10,177	1/mes	10,170

Taula 8.6. Nº de dies en funcionament del sistema (alternativa 1) vs freqüència de captura, resolució i qualitat de la imatge (50).

640x480 (50%)	140.000	1280x720 (50%)	370.000	3280x2464 (50%)	3.130.000
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	-	1/70s	-	1/70s	-
1/90s	-	1/90s	-	1/90s	-
1/3min	4,602	1/3min	-	1/3min	-
1/5min	5,893	1/5min	-	1/5min	-
1/6min	6,467	1/6min	4,602	1/6min	-
1/8min	7,087	1/8min	5,438	1/8min	-
1/10min	7,464	1/10min	6,486	1/10min	-
1/30min	9,077	1/30min	8,105	1/30min	-
1/55min	9,546	1/55min	8,931	1/55min	4,603
1/1h	9,595	1/1h	9,923	1/1h	5,486
1/10h	10,115	1/10h	10,048	1/10h	9,160
1/15h	10,136	1/15h	10,090	1/15h	9,475
1/dia	10,1511	1/dia	10,122	1/dia	9,727
1/5dies	10,171	1/5dies	10,166	1/5dies	10,083
1/10dies	10,174	1/10dies	10,171	1/10dies	10,129
1/mes	10,177	1/mes	10,172	1/mes	10,394

En aquest cas, mitjançant aquest nou sistema, es poden aconseguir fins a 10 dies de funcionament.

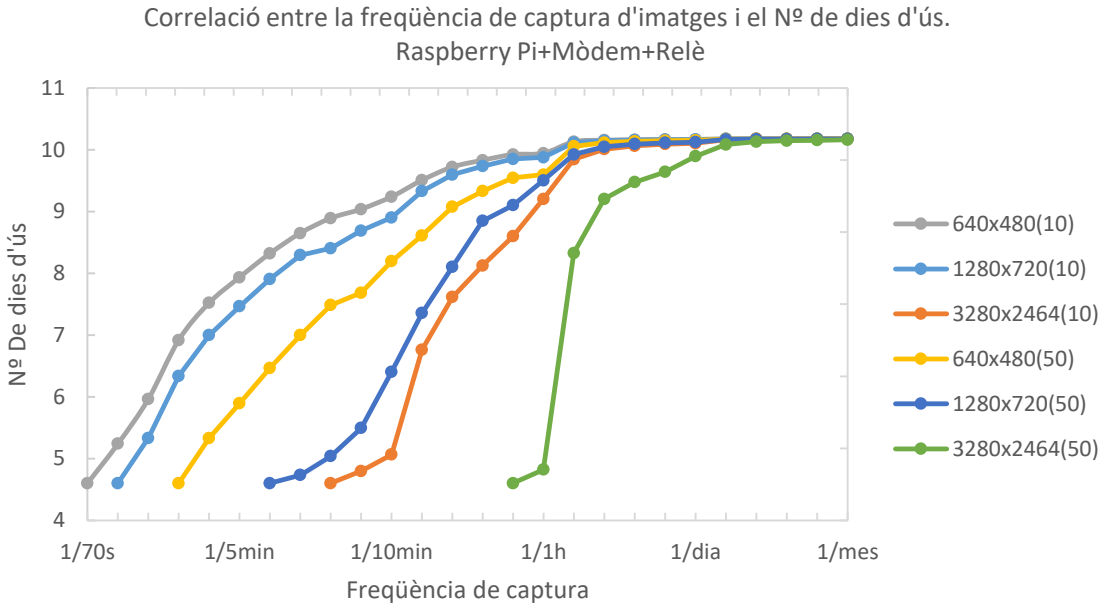

Figura 8.17. Correlació entre el número de dies d'ús i el número de fotos enviades de l'alternativa 1.

En la figura 8.17 es mostra la correlació entre el número de dies de funcionament de l'alternativa 1 amb respecte la freqüència de captura de les imatges, així com respecte la resolució de la imatge i la qualitat. De la mateixa forma que en el sistema dissenyat, quant major és la imatge (major resolució i major qualitat) menor número d'imatges es poden enviar per dia.

- **2na alternativa, BeagleBone Black + utilització d'un relé d'estat sòlid:** finalment, una altra alternativa seria la utilització d'un altre tipus de computadora de placa reduïda com és per exemple la BeagleBone Black. Tot i tenir consums energètics pràcticament idèntics, tant a nivell d'estrès de càrrega de la CPU, com inclús amb el seu propi mòdul de càmera, amb respecte la Raspberry Pi, aquest model incorpora l'opció de "Sleep-Mode", és a dir, permet desconnectar tots els seus perifèrics, així com totes les opcions innecessàries per tal de minimitzar el seu consum energètic. Tot i que el datasheet d'aquest dispositiu no indica exactament el consum energètic en el mode "Sleep-Mode", aquest s'aproxima a 50 mA. Per la realització dels càlculs de consum energètic, es tindrà en compte un valor aproximat de 40 mA pel consum de la BeagleBone amb la càmera (indicat en el datasheet [17]).

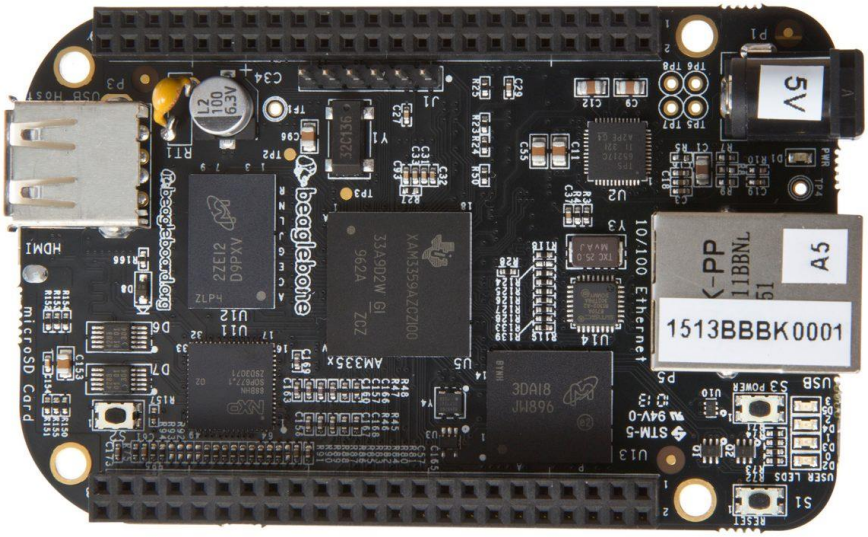

Figura 8.18. BeagleBone Black. [17]

A partir d'aquestes aproximacions, es poden realitzar els càlculs dels temps de funcionament del sistema a través del consum energètic en transmissió d'imatges de 4,642 W, corresponent a la suma del consum energètic del mòdem en transmissió de 2,642 W i el consum de la BeagleBone Black amb la càmera connectada de 0,4 A·5 V = 2 W, i el consum energètic en repòs de 5 V· 0,05 A = 0,25 W.

Taula 8.7. Nº de dies en funcionament del sistema (alternativa 2) vs freqüència de captura, resolució i qualitat de la imatge (10).

640x480 (10%)	50.000	1280x720 (10%)	70.000	3280x2464 (10%)	480.000
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	4,782	1/70s	-	1/70s	-
1/90s	6,055	1/90s	4,782	1/90s	-
1/3min	11,384	1/3min	9,076	1/3min	-
1/5min	17,414	1/5min	14,162	1/5min	-
1/6min	20,962	1/6min	17,206	1/6min	-
1/8min	28,483	1/8min	24,335	1/8min	4,782
1/10min	37,595	1/10min	33,038	1/10min	5,677
1/30min	52,756	1/30min	47,274	1/30min	15,102
1/55min	64,693	1/55min	60,0355	1/55min	24,250
1/1h	74,493	1/1h	70,395	1/1h	37,485
1/10h	85,866	1/10h	85,064	1/10h	71,382
1/15h	86,822	1/15h	86,274	1/15h	76,376
1/dia	87,554	1/dia	87,204	1/dia	83,499
1/5dies	88,548	1/5dies	88,476	1/5dies	87,030
1/10dies	88,673	1/10dies	88,637	1/10dies	87,906
1/mes	88,756	1/mes	88,746	1/mes	88,500

Taula 8.8. Nº de dies en funcionament del sistema (alternativa 2) vs freqüència de captura, resolució i qualitat de la imatge (50).

640x480 (50%)	140.000	1280x720 (50%)	370.000	3280x2464 (50%)	3.130.000
Freqüència captura	**Temps d´ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**	**Freqüència captura**	**Temps d'ús (dies)**
1/70s	-	1/70s	-	1/70s	-
1/90s	-	1/90s	-	1/90s	-
1/3min	4,782	1/3min	-	1/3min	-
1/5min	7,694	1/5min	-	1/5min	-
1/6min	9,526	1/6min	4,782	1/6min	-
1/8min	12,134	1/8min	6,187	1/8min	-
1/10min	18,938	1/10min	7,322	1/10min	-
1/30min	32,211	1/30min	18,858	1/30min	-
1/55min	45,346	1/55min	33,039	1/55min	4,782
1/1h	62,935	1/1h	45,938	1/1h	4,999
1/10h	81,629	1/10h	75,398	1/10h	34,012
1/15h	83,887	1/15h	79,029	1/15h	42,824
1/dia	85,664	1/dia	85,384	1/dia	57,398
1/5dies	88,157	1/5dies	87,448	1/5dies	70,394
1/10dies	88,476	1/10dies	88,120	1/10dies	83,216
1/mes	88,689	1/mes	88,572	1/mes	86,857

Com es pot veure en les taules 8.7 i 8.8, mitjançant aquesta última alternativa, el temps de funcionament del sistema es veu altament incrementat degut a la utilització del "Sleep Mode" de la BeagleBone Black, la qual minimitza de forma dràstica el consum energètic del mòdem en repòs, aconseguint temps de funcionament fins a pràcticament 3 mesos.

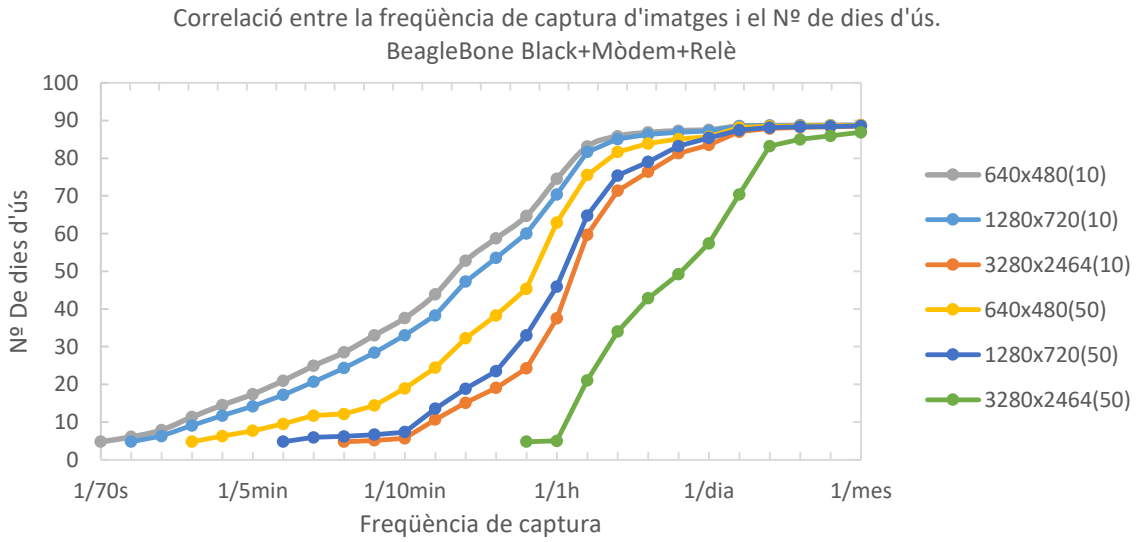

Figura 8.19. Correlació entre el número de dies d'ús i el número de fotos enviades de l'alternativa 2.

Com en els altres dissenys, el temps de funcionament del sistema disminueix quant major número d'imatges s'envien, així com quant major és la mida de la imatge (veure figura 8.19).

8.4.3. Comparativa d'alternatives

Finalment doncs, es pot concloure que el sistema dissenyat pot ser útil en aplicacions de curta durada de funcionament, no obstant això, per objectius de mitja-llarga durada seria convenient implementar alguna de les altres dues alternatives mencionades anteriorment.

En la figura 8.20 es mostra una correlació entre el número de dies de funcionament de les diferents alternatives, així com respecte el número de fotos enviades respecte el temps, amb imatges d'una resolució de 640x480 i un qualitat d'imatge d'un 10%. La diferència entre la primera alternativa i el sistema original no és gaire, d'aproximadament 4 dies de diferència, no obstant això, amb respecte la segona alternativa com es pot apreciar, la diferència és molt significativa.

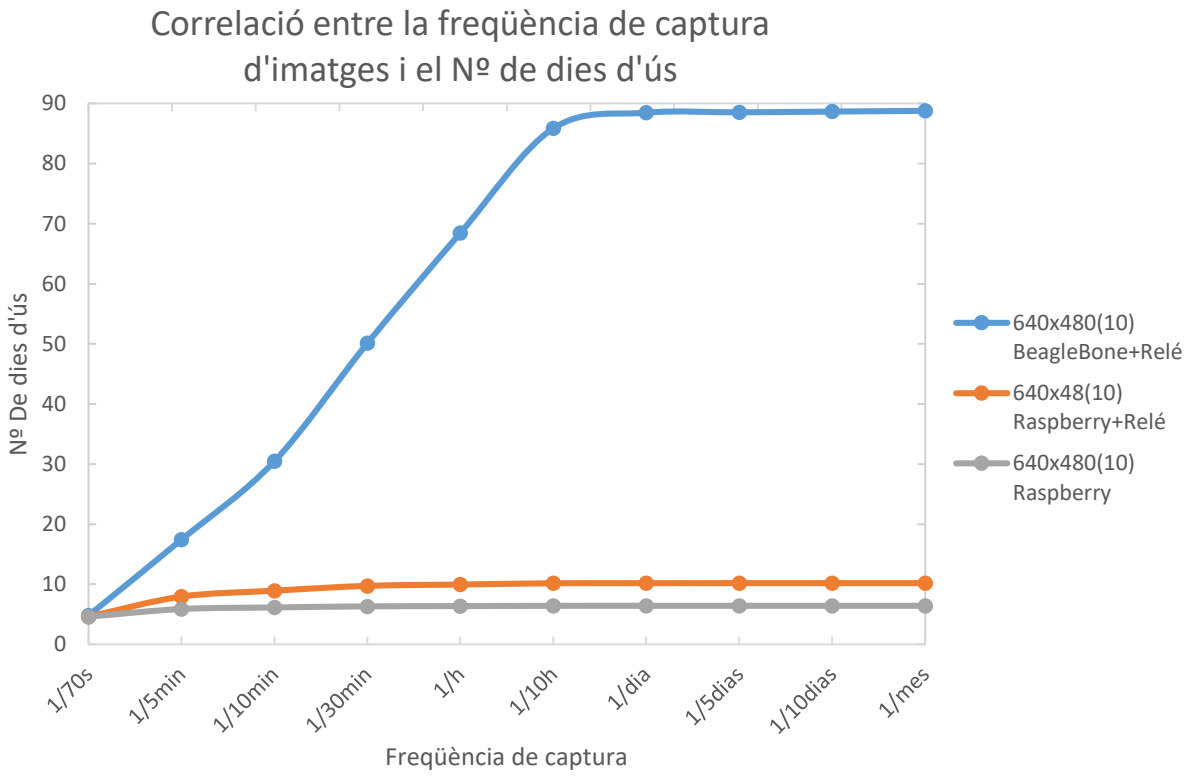

Figura 8.20. Correlació entre el número de dies de funcionament i la freqüència de captura d'imatges respecte les diferents alternatives.

9. Anàlisi de l'impacte ambiental

Actualment, en qualsevol projecte de caire industrial és imprescindible realitzar una valoració de l'impacte ambiental del sistema dissenyat, per tal de prendre consciència de la seva influència en el medi ambient, així com per estudiar possibles millores en el disseny i la selecció dels components per tal de causar el mínim dany possible en el medi.

El sistema dissenyat en aquest projecte està pensat per ser utilitzat en el medi aquàtic. Prèviament a la seva utilització, cal assegurar-se de la ubicació del mateix, per tal d'evitar possibles danys a la fauna marina, així com possibles afectacions als sers vius marins. Respecte als components utilitzats que composen al sistema, no suposen cap perill ni per l'entorn ni pels usuaris que manipulin el sistema.

Un altre aspecte a tenir en compte en aquest projecte en concret, és la potencial contaminació acústica, és a dir, l'efecte de la freqüència dels senyals acústics emesos pels mòdems, els quals si fossin gaire altes, podrien afectar a certes espècies marines com són per exemple els cetacis, els quals utilitzen les ones sonores per tal de comunicar-se. Els mòdems d'aquest projecte utilitzen un rang de freqüències no gaire altes (18-34 kHz) i poca potència d'emissió, les quals no afecten negativament a cap espècie marina (per exemple, els cetacis emeten habitualment senyals en el rang de pocs Hz).

Cal assegurar-se que un cop les bateries del sistema s'han esgotat, aquestes es reciclen de forma adequada, ja que el liti és un metall alcalí força contaminant. Pel que fa als demes components electrònics, aquests poden contenir elements tòxics pel medi. Per tant, en cas d'avaries o funcionaments incorrectes, aquests s'han de tractar de forma adequada portant-los als punts adients de recollida d'aquests materials.

A més, cal destacar, que la vida útil dels components electrònics utilitzats és molt elevada, superant amb diferència el temps el qual el sistema estarà en funcionament, per tant, en utilitzar un encapsulat basat en un tub acrílic, el sistema es pot recuperar, i per tant els components electrònics es poden reutilitzar, reduint així l'excés de residus.

Per tant, es pot concloure que el prototip dissenyat té un impacte molt reduït sobre el medi ambient, sempre i quan es tinguin en compte les consideracions indicades anteriorment. El consum energètic del sistema és molt baix i la única contaminació que pot generar és la deguda a les pertorbacions electromagnètiques dels reguladors de tensió, encara que aquestes són pràcticament nul·les.

Conclusions

Aquest projecte ha permès el desenvolupament de coneixements d'àmbits relacionats amb l'especialitat cursada, l'electrònica industrial, els quals no s'han arribat a tractar al llarg de la carrera, com és la comunicació acústica submarina.

S'ha pogut combinar en el mateix projecte, diferents disciplines com són l'electrònica, per tal de dur a terme el disseny del sistema i la selecció dels components necessaris, la informàtica, per tal de programar la Raspberry Pi, i finalment, l'àrea de comunicacions, per tal d'interconnectar la Raspberry i l'ordinador amb els mòdems acústics, així com la comunicació entre els mòdems.

El disseny del sistema ha estat realitzat amb èxit, tot i que no s'ha pogut arribar a implementar de forma completa el model dissenyat físicament, degut a la demora en les entregues, així com pels costos econòmics.

Respecte al funcionament del sistema, aquest ha pogut ser comprovat amb èxit mitjançant els mòdems virtuals de l'empresa que fabrica els mòdems acústics, així com amb els propis mòdems físics fent proves dins d'un aquari.

Tot i que les proves realitzades amb els mòdems físics no han resultat molt satisfactòries degut als rebots de les parets de l'aquari, s'ha pogut realitzar una estimació de la viabilitat a través de les proves realitzades amb els mòdems virtuals.

Alguna de les propostes d'ampliació del projecte podrien ser la realització del mateix sistema amb algunes de les alternatives mencionades, per tal de dur a terme una comparativa a nivell d'estalvi energètic. També, per tal d'incrementar la fiabilitat de les dades, les proves s'haurien de realitzar a mar ober

Pressupost i Anàlisi Econòmica

En qualsevol projecte en el qual la finalitat és el disseny d'un prototip o sistema en l'àmbit de l'enginyeria és essencial realitzar un pressupost i una valoració econòmica del mateix, per tal d'analitzar la viabilitat del sistema, detectar possibles sobrecostos, així com, en cas de què es tracti d'un sistema destinat a ser comercialitzat, estudiar els potencials competidors.

El pressupost econòmic s'ha desglossat en 3 costos: materials, indirectes i d'enginyeria.

- **Costos materials:** s'han tingut en compte els components essencials que composen el sistema de visió submarina.

Taula 1. Costos materials del projecte.

Descripció	Quantitat	Preu
Raspberry Pi 2 model B	1	31,90 €
Camera Module V2	1	23,09 €
MicroSD SAMSUNG EVOPlus 32 GB	1	14,99 €
EvoLogics S2CM 18/34	2	7.500 €
BlueRobotics 14,8 V-18 Ah	2	221,56 €
Step-down LM2596	2	1,45 €
Encapsulament BlueRobotics 6''	1	258,04 €
TOTAL	**15.774,04 €**	

- **Costos indirectes:** en aquest bloc s'han considerat tots els costos referents a llicències de programes utilitzats per la realització del projecte, així com materials que no composen de forma implícita el prototip dissenyat.

Taula 2. Costos indirectes del projecte.

Descripció	Quantitat	Preu
Llicència LabView 2018	1	3.451,00 €
Llicència Spyder (Llenguatge Python)	1	0,00 €
Portàtil de sobretaula	1	500,00 €
TOTAL	**3.951,00 €**	

- **Costos d'enginyeria:** per tal de tenir en compte els costos associats a les hores dedicades per la realització del prototip, s'han dividit en cinc blocs diferents: selecció dels components i muntatge, configuració del sistema, programació del software, estudi de consums energètics i finalment la redacció de la memòria.

Taula 3. Costos d'enginyeria del projecte.

Descripció	Hores	Preu/hora	Preu
Selecció dels components i muntatge	100	8,00 €	800,00 €
Configuració del sistema	80	7,00 €	560,00 €
Programació del Software	250	12,00 €	3.000,00 €
Estudi de consums energètics	80	8,00 €	160,00 €
Redacció de la memòria	100	5,00 €	500,00 €
TOTAL		5.020,00 €	

Finalment, tal i com s'observa en la taula 4, el cost total de la realització del sistema de visió submarina és de 24.715,04 €.

Taula 4. Cost total del projecte.

Blocs	Preu
Cost de components	15.774,04 €
Costos indirectes	3.951,00 €
Costos d'enginyeria	5.020,00 €
TOTAL	24.715,04 €

El cost pot semblar molt elevat, no obstant això cal tenir en compte el gran valor econòmic dels mòdems acústics S2CM 18/34 d'EvoLogics (15.000 € els dos) i el cost de la llicència del programa LabView, la qual si es contactes amb l'empresa de National Instruments, es podria aconseguir una versió d'estudiant, ja que es tracta d'un projecte universitari.

Per tant, tot i ser una inversió econòmica important, el sistema ha estat dissenyat per tal de minimitzar els costos del projecte, evitar possibles sobrecostos i així reduir al màxim la competència en relació a prototips que disposin de funcionalitats similars.

Bibliografia

[1] J.Y. Khan. (2011). *Short-Range Underwater Acoustic Communication Networks.* DOI: 10.5772/24098. [en línia]. Disponible en:
https://www.researchgate.net/publication/221918253_ShortRange_Underwater_Acoustic_Communication_Networks

[2] TELEDYNE MARINE. *Acoustic Modems.* (n.d.). [en línea]. Disponible en:
http://www.teledynemarine.com/acoustic-modems

[3] AquaSeNT. (n.d.). *Datasheet AM-D2000.* [en línea]. Disponible en:
https://static1.squarespace.com/static/53f382fae4b05ea0cfc56833/t/559aac06e4b00d130d6042f0/1436199942842/AM-D2000.pdf

[4] Sendra, Sandra. Lloret, Jaime. Jimenez, Jose. Parra, Lorena. (2016). *Underwater Acoustic Modems.* DOI: 10.1109/JSEN.2015.2434890. [en línea]. Disponible en:
https://ieeexplore.ieee.org/abstract/document/7113785/keywords#keywords

[5] Raspberry Pi Foundation. (n.d.). *Documentation*. [en línea]. Disponible en:
https://www.raspberrypi.org/documentation/hardware/raspberrypi/

[6] EvoLogics.de. (n.d.). *S2CM 18/34 Underwater Acoustic Modem.* [en línea]. Disponible en:
https://evologics.de/acoustic-modem/18-34/m-serie

[7] M. Sozer, Ethem. Stojanovic, Milicia. G. Proakis, John. (2000). *Underwater Acoustic Netowkrs*. DOI: 10.1109/48.820738. [en línea]. Disponible en:
https://ieeexplore.ieee.org/document/820738

[8] BlueRobotics. (n.d.). *Lithium-ion Battery (14,8 V, 18 AH).* [en línea]. Disponible en:
https://www.bluerobotics.com/store/comm-control-power/batteries/battery-li-4s-18ah-r2-rp/

[9] Texas Instruments. (n.d.). *Datasheet LM2596 Simple Switcher Power Converter 150 kHz 3 A Step Down Voltage Regulator.* [en línea]. Disponible en:
http://www.ti.com/lit/ds/symlink/lm2596.pdf

[10] BlueRobotics. (n.d.). *Watertight Enclosure for ROV/AUV (6'' Series).* [en línea]. Disponible en:

https://www.bluerobotics.com/store/watertight-enclosures/6-series/wte6-asm-r1/

[11] Python Software Foundation. (n.d.). *Low Level-Networking interface.* [en línea]. Disponible en: https://docs.python.org/2/library/socket.html

[12] EvoLogics.de. (2017). *S2C Reference Manual Edition Networking Firmware Version 1.9.* [en línea].

[13] EvoLogics.de. (2017). *EvoLogics DMAC Emulator User Guide.* [en línea].

[14] National Instruments. (n.d.). *USER GUIDE NI USB-6008/6009.* [en línea]. Disponible en: http://www.ni.com/pdf/manuals/371303n.pdf

[15] IXYS. (n.d.). *Datasheet CPC1114N Optomos Relay.* [en línea]. Disponible en: http://www.ixysic.com/home/pdfs.nsf/www/CPC1114N.pdf/$file/CPC1114N.pdf

[16] Li, Baosheng. Zhou, Shengli. Stojanovic, Milica. Freitag, Lee. Willet, Peter. (2008). *Multicarrier Communication Over Underwater Acoustic Channels With Nonuniform Doppler Shifts.* DOI: 10.1109/JOE.2008.920471. [en línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/4554200

[17] Coley, Gerard. (2013). *BeagleBone Black System Reference Manual.* [en línea]. Disponible en: https://cdn-shop.adafruit.com/datasheets/BBB_SRM.pdf

Annex A

A1. Programa versió 1: Enviament d'imatges automàtic

```
2   # -*- coding: utf-8 -*-
3   """
4   Created on Mon Feb 18 23:27:35 2019
5
6   @author: Oriol Solé
7
8   Programa versió1: Enviament d'imatges automàtic (Programa modem
    transmissor)
9   """
10
11  #Llibreries#################################################################
12
13  #Importació de les llibreries adients:
14  import base64   #Llibreria per codificar i descodificar les imatges en
15  #base 64
16  import socket   #Llibreria per fer ús dels sockets
17  import time     #Llibreria per manipular tasques relacionades amb temps
18  import datetime #Llibreria per obtenir dates
19  import os       #Llibreria per manipular paths
20  import numpy    #Llibreria per executar operacions estadístiques
21  import xlsxwriter #Llibreria per operar amb fitxers Excel
22  from picamera import PiCamera #Llibreria per controlar la càmera de la
23  #Raspberry
24
25  #Definició Mòdems###########################################################
26
27  #Definició de les IPs i dels ports de comunicació dels mòdems acústics
28  #físics i virtuals transmissors
29  HOSTNAME1EM = '10.42.57.2'
30  HOSTNAME1 = '192.168.1.196'
31  PORT1 = 9200
32
33  #Classe funció sockets######################################################
34
35  #Creació de la classe netcat per ubicar les funcions relatives a la
36  #manipulació dels sockets
37  class netcat(object):
38      #Definició de la funció inicial per la configuració del mòdem acústic
```

```
39      def __init__ (self,hostname,port,modem_name):
40          #Temps d'espera per assegurar la connectivitat ethernet de la
41          #Raspberry al iniciar-se
42          time.sleep(10)
43          #Establiment de la ip, el port de comunicació i el nom del mòdem
44          self.ip = hostname
45          self.port = port
46          self.name = modem_name
47          #Creació del socket i connectivitat
48          try:
49              self.netcat = socket.socket(socket.AF_INET,
50              socket.SOCK_STREAM)
51              self.netcat.settimeout(0.1)
52              self.netcat.connect((self.ip, self.port))
53          except:
54              print "No s'ha pogut establir connexio"
55
56          #Configuracions inicials:
57          print 'Activacio permissos'
58          self.send('AT@CTRL')
59          print 'Desactivacio notificacions extesses'
60          self.send('AT@ZX0')
61          print 'Desactivacio notificacions de posicionament'
62          self.send('AT@ZU0')
63          print 'Reiniciament del buffer de transmissio:'
64          self.send('ATZ4')
65          print 'Configuracio del numero de bytes del buffer de
66          transmissio:'
67          self.send('AT@ZL1000000')
68          print 'Guardar en memoria les configuracions del modem'
69          self.send('AT&W')
70
71      #Funció de lectura de dades
72      def port_read(self):
73          buff = ''
74          read = ''
75          while 1:
76              #Bloc d'excepció, en cas de no poder establir connectivitat,
77              #i que no es tracti d'un error de timeout, es torna a intentar
78              #restablir la connexió perduda
79              try:
80                  read = self.netcat.recv(1)
81              except socket.error as error:
82                  if str(error)!= "timed out":
```

```
83                      try:
84                          self.netcat = socket.socket(socket.AF_INET,
85                          socket.SOCK_STREAM)
86                          self.netcat.settimeout(0.1)
87                          self.netcat.connect((self.ip, self.port))
88                          print "Connexio recuperada"
89                      except:
90                          print "Intent de connexio fallit"
91                          time.sleep(2)
92                          pass
93                  break
94              if read:
95                buff += read
96              else:
97                  break
98          return buff
99
100     #Funció per escriptura de comandes AT
101     def port_write(self,command):
102         while 1:
103             try:
104                 print 'Send',self.name +':',command
105                 self.netcat.sendall(command + '\n')
106                 break
107             except:
108                 try:
109                     self.netcat = socket.socket(socket.AF_INET,
110                     socket.SOCK_STREAM)
111                     self.netcat.settimeout(0.1)
112                     self.netcat.connect((self.ip, self.port))
113                     print "Connexio recuperada"
114                 except:
115                     print "Intent de connexio fallit"
116                     time.sleep(2)
117                     pass
118         return
119
120     #Funció d'enviament de comandes AT amb comprovació de resposta
121     def send(self,command):
122         while 1:
123             self.port_write(command)
124             read = self.port_read()
125             x1 = read.find("BUSY CLOSING CONNECTION")
126             x2 = read.find("FAILED")
```

```
127             x3 = read.find("OK")
128             if x1!=-1:
129                 print "Received", self.name + ':', "FAILED"
130             elif x2!=-1:
131                 print "Received",self.name + ':', "BUSY CLOSING
132                 CONNECTION"
133             elif x3!=-1:
134                 print "Received",self.name + ':', "OK"
135                 break
136             elif read == "":
137                 print "buit"
138             else:
139                 break
140             time.sleep(2)
141         return read
142 
143     #Funció per el tancament del socket en cas d'interrupció del programa
144     def close(self):
145         self.netcat.shutdown(socket.SHUT_WR)
146         self.netcat.shutdown(socket.SHUT_RD)
147         self.netcat.close()
148 
149 #Funcions#########################################################
150 
151 #Funció per separar la imatge en un cert numero de bytes fora de la
152 #classe
153 def paquets(foto,longitud):
154     return [foto[i:i + longitud] for i in range (0,len(foto), longitud)]
155 
156 #Principal########################################################
157 
158 #Definició del mòdem amb la IP, el port de comunicació i el nom (crida
    de la
159 #classe netcat)
160 modem2 = netcat(HOSTNAME1,PORT1,'modem2')
161 
162 #Obtenció del path on s'ubica el programa
163 dir = os.path.dirname(os.path.abspath(__file__))
164 
165 #Definició variables inicials
166 llistatemps = []
167 llistatemps_aux = []
168 llistamissatges = []
169 llistafotos = []
```

```
img_correctes = 0
img=0

#Inicialització de la càmera i establiment de la resolució i qualitat
camera = PiCamera()
camera.resolution=(640,480)
qualitat = 10

#Bucle infinit principal
while 1:
    #Creació de directoris segons els dies
    save_path1 = dir+'/'+str(datetime.date.today())

    try:
        os.stat(save_path1)
    except:
        os.mkdir(save_path1)
        llistatemps = []
        llistatemps_aux = []
        llistamissatge = []
        llistafotos = []
        img=0
        img_correctes = 0

    #Captació del temps de inici de captura de imatges
    temps=time.time()
    #Captura de la imatge
    camera.capture(save_path1+'/'+str(temps)+'.jpg',quality=qualitat)
    #Codificació de la imatge en base64
    with open(save_path1+'/'+str(temps)+'.jpg',"rb") as imageFile:
        imatge = base64.b64encode(imageFile.read())

    #Creació de arxius Excel amb informació relativa al enviament de
    #imatges
    workbook=xlsxwriter.Workbook(save_path1+'/fitxerEstadistic.xlsx')
    worksheet = workbook.add_worksheet()
    row = 1
    col = 0

    bold = workbook.add_format({'bold': True})
    red = workbook.add_format({'color':'red'})

    worksheet.write(0,0,'Nom Imatge',bold)
    worksheet.write(0,1,'Temps en (s)',bold)
```

```
214     worksheet.write(0,2,'Estat Imatge',bold)
215     worksheet.write(0,3,'Temps mitja en (s)',bold)
216     worksheet.write(0,4,'Desviacio estandard',bold)
217     worksheet.write(0,5,'Percentatge de fotos enviades',bold)
218     worksheet.write(0,6,'Numero de mostres',bold)
219
220     #Obtenció de la mida de la imatge, i creació de paquets de 1024 bytes
221     mida = len(imatge)
222     nimatge = paquets(imatge,1024)
223
224     #Generació d'un fitxer amb possibles errors
225     fitxer = os.path.join(save_path1,"MissatgeError.txt")
226
227     #Enviament del numero de bytes de la imatge a transmetre
228     resposta = modem2.send('AT*SEND,'+str(len(str(mida))+1)+',3,'+
229     str(mida)+'#')
230
231     #Comprovació de la correcte recepció del numero de bytes
232     t = 0
233     while 1:
234         resposta2 = modem2.port_read()
235         x=resposta2.find("OK")
236         if x!= -1:
237             break
238         else:
239             t = t+1
240             if t == 100:
241                 break;
242
243     #Si el mòdem respon, envia la imatge amb un flag final de exclamació
244     if t != 100:
245         i = 0
246         while i<len(nimatge):
247             if i == len(nimatge)-1:
248                resposta3=modem2.send('AT*SEND,'+str(len(nimatge[i])+1)+
249                ',3,'+str(nimatge[i]+"!"))
250             else:
251                resposta3=modem2.send('AT*SEND,'+str(len(nimatge[i]))+
252                ',3,'+str(nimatge[i]))
253             i = i+1
254
255         #Un cop es enviada la imatge s'espera la confirmació de la
256         #recepció del n
257         #"OK" en cas d'haver sigut correctament enviada, "KO" en cas d'un
```

```
258         #enviament erroni o, si el mòdem receptor es desconnecta,
259         #"Disconnect"
260         t = 0
261         aux1=1
262         missatge = ''
263         while 1:
264             resposta = modem2.port_read()
265             x=resposta.find("OK")
266             x1=resposta.find("KO")
267             #Si la resposta es correcta, s'elimina la imatge de la
268             #Raspberry
269             if x!= -1:
270                 os.system('rm '+save_path1+'/'+str(temps)+'.jpg')
271                 img_correctes = img_correctes+1
272                 missatge = "OK"
273                 tempsfinal = time.time()-temps
274                 llistatemps.append(tempsfinal)
275                 llistatemps_aux.append(tempsfinal)
276                 break
277             #En cas d'incorrecte recepció, s'emmagatzema la imatge, i es
278             #guarden els missatges d'error
279             else:
280                 if x1!=-1:
281                     try:
282                         os.stat(save_path1)
283                     except:
284                         os.mkdir(save_path1)
285
286                     missatge = "KO"
287                     llistatemps_aux.append("")
288                     fh=open(fitxer,"a")
289                     fh.write(str(temps)+': '+'El numero de bytes de la'+
290                     'imatge rebuda no coincideix amb la original'+'\n')
291                     fh.close
292                     break
293                 #Si no s'obté resposta, es comprova l'estat de
294                 #comunicació dels
295                 #mòdems acústics i es finalitza l'enviament, reiniciant
296                 #el buffer de transmissió
297                 else:
298                     resposta = modem2.send("AT?S")
299                     aux1 =1
300                     x = resposta.find("INITIATION LISTEN")
301                     x2= resposta.find("1000000 bytes free")
```

```
302                     #Si es detecta que una pèrdua de la connexió, degut
303                     #a un microtall d'alimentació o de connexió ethernet
304                     #s'envia el flag "$" per informar al usuari que deixi
305                     #de llegir i esperi la nova recepció del numero de
306                     #bytes
307                     if x!=-1:
308                         aux1 =0
309                         modem2.send('AT*SEND,1,3,$')
310                         time.sleep(3)
311                         modem2.send("ATZ4")
312                     if x2!=-1:
313                         t=t+1
314                     if t == 50 or aux1 == 0:
315                         try:
316                             os.stat(save_path1)
317                         except:
318                             os.mkdir(save_path1)
319 
320                         missatge = "Disc."
321                         llistatemps_aux.append("")
322 
323                         fh=open(fitxer,"a")
324                         fh.write(str(temps)+': '+'El client es va'+
325                         'desconnectar mentres la imatge era enviada'+
326                         '\n')
327                         fh.close
328 
329                         break;
330 
331     #Si el client no contesta al cap d'un temps "t", es guarda la imatge
332     #i es genera un missatge d'error en el fitxer
333     else:
334         missatge = "OFF"
335         llistatemps_aux.append("")
336         try:
337             os.stat(save_path1)
338         except:
339             os.mkdir(save_path1)
340 
341         fh=open(fitxer,"a")
342         fh.write(str(temps)+': '+'El client estava desconnectat'+'\n')
343         fh.close
344 
345     #S'omple l'Excel amb les dades
```

```
346     img=img+1
347     llistamissatges.append(missatge)
348     llistafotos.append(temps)
349
350     if len(llistatemps)==0:
351         temps_mitja = ""
352         desviacio_estandard = ""
353         percentatge = "0"
354     else:
355         temps_mitja = sum(llistatemps)/len(llistatemps)
356         desviacio_estandard = numpy.std(llistatemps)
357         percentatge = (img_correctes*100)/img
358
359
360     worksheet.write(1,3,temps_mitja)
361     worksheet.write(1,4,desviacio_estandard)
362     worksheet.write(1,5,percentatge)
363     worksheet.write(1,6,img)
364
365     row = 1
366
367     for temps in (llistatemps_aux):
368         worksheet.write(row,1,temps)
369         row+=1
370
371     row = 1
372     i=0
373
374     for missatges in (llistamissatges):
375         x1 = missatges.find("OK")
376         if x1!=-1:
377             worksheet.write(row,2,missatges)
378             worksheet.write(row,0,llistafotos[i])
379         else:
380             worksheet.write(row,2,missatges,red)
381             worksheet.write(row,0,llistafotos[i],red)
382         row+=1
383         i+=1
384
385     row = 1
386     workbook.close()
387
388 modem2.close()
```

A2. Programa versió 1: Recepció d'imatges automàtic

```
# -*- coding: utf-8 -*-
"""
Created on Sat Feb 16 13:40:41 2019

@author: Oriol Solé

Programa versió1: Enviament d'imatges automàtic (Programa modem receptor)
"""

#Llibreries###################################################################

#Importació de les llibreries adients:
import base64   #Llibreria per codificar i descodificar les imatges
#en base 64
import socket   #Llibreria per fer ús dels sockets
import time     #Llibreria per manipular tasques relacionades amb
#temps
import datetime #Llibreria per obtenir dates
import os       #Llibreria per manipular paths
import sys      #Llibreria per executar instruccions amb permís del
#sistema

#Configuració Mòdems##########################################################

#Definició de les IPs i dels ports de comunicació dels mòdems
#acústics físics i virtuals receptors

HOSTNAME2EM = '10.42.57.3'
HOSTNAME2 = '192.168.1.197'
PORT2 = 9200

#Classe funció sockets########################################################

#Creació de la classe netcat per ubicar les funcions relatives a la
#manipulació dels sockets
class netcat(object):
#Definició de la funció inicial per la configuració del mòdem acústic
  def __init__ (self,hostname,port,modem_name):
        #Establiment de la ip, el port de comunicació i el nom del
        #mòdem
        self.ip = hostname
        self.port = port
        self.name = modem_name
        #Creació del socket i connectivitat
        try:
            self.netcat = socket.socket(socket.AF_INET,
            socket.SOCK_STREAM)
            self.netcat.settimeout(0.1)
            self.netcat.connect((self.ip, self.port))
        except:
            print ("El programa es tancara, ja que no s'ha pogut" +
            "establir una connexió degut a que el sistema no està" +
            "alimentat, connecti la alimentacio i executi de nou" +
            "el programa")
```

```
56              time.sleep(5)
57              sys.exit(1)
58
59          #Configuracions inicials:
60          print 'Activacio permissos'
61          self.send('AT@CTRL')
62          print 'Desactivacio notificacions extesses'
63          self.send('AT@ZX0')
64          print 'Desactivacio notificacions de posicionament'
65          self.send('AT@ZU0')
66          print 'Reiniciament del buffer de transmissio:'
67          self.send('ATZ4')
68          print 'Configuracio del numero de bytes del buffer de' +
69          'transmissio:'
70          self.send('AT@ZL1000000')
71          print 'Guardar en memoria les configuracions del modem'
72          self.send('AT&W')
73
74      #Funció de lectura de dades
75      def port_read(self):
76          buff = ''
77          read = ""
78          while 1:
79              #Bloc d'excepció, en cas de no poder establir
80              #connectivitat, i que
81              #no es tracti d'un error de timeout, es torna a intentar
82              #restablir
83              #la connexió perduda
84              try:
85                  read = self.netcat.recv(1)
86              except socket.error as error:
87                  if str(error) != "timed out":
88                      print ("El programa es tancara degut a un" +
89                      "problema amb la alimentacio, comprovala i" +
90                      "executa de nou el programa")
91                      time.sleep(5)
92                      sys.exit(1)
93                      break
94                  break
95              if read:
96                  buff += read
97              else:
98                  break
99          return buff
100
101     #Funció per escriptura de comandes AT
102     def port_write(self,command):
103       while 1:
104          print 'Send',self.name +':',command
105          try:
106              self.netcat.sendall(command + '\n')
107              break
108          except:
109              try:
110                  self.netcat = socket.socket(socket.AF_INET,
111                  socket.SOCK_STREAM)
112                  self.netcat.settimeout(0.1)
113                  self.netcat.connect((self.ip, self.port))
```

```
114                 print "Connexio recuperada"
115             except:
116                 print "Intent de connexio fallit"
117                 time.sleep(2)
118                 pass
119     return
120 
121   #Funció d'enviament de comandes AT amb comprovació de resposta
122   def send(self,command):
123      while 1:
124             self.port_write(command)
125             read = self.port_read()
126             x1 = read.find("BUSY CLOSING CONNECTION")
127             x2 = read.find("FAILED")
128             x3 = read.find("OK")
129             if x1!=-1:
130                 print "Received", self.name + ':', "FAILED"
131             elif x2!=-1:
132                 print "Received",self.name + ':', "BUSY CLOSING"+
133                 "CONNECTION"
134             elif x3!=-1:
135                 print "Received",self.name + ':', "OK"
136                 break
137             elif read == "":
138                 print 'buit'
139                 time.sleep(2)
140             else:
141                 break
142             time.sleep(2)
143      return read
144 
145   #Funció per el tancament del socket en cas d'interrupció del
    programa
146   def close(self):
147     self.netcat.shutdown(socket.SHUT_RD)
148     self.netcat.close()
149 
150 
151 #Principal#################################################################
152 
153 #Definició del mòdem amb la IP, el port de comunicació i el nom
154 #(crida de la classe netcat)
155 modem3 = netcat(HOSTNAME2,PORT2,'modem3')
156 
157 #Bucle infinit principal
158 while 1:
159     numbytesaux = ""
160     #Bucle per l'espera del numero de bytes a rebre amb el flag "#"
161     while 1:
162         numbytesaux = modem3.port_read()
163         z = numbytesaux.find("#")
164         if z!= -1:
165             numbytesaux = numbytesaux.replace("#","")
166             tempsImatge = time.time()
167             break
168 
169     numbytesaux1 = numbytesaux.split(",")
170     numbytes = int(numbytesaux1[-1])
```

```
171     print "El numero de bytes de la imatge son: "+str(numbytes)
172
173     #Es contesta si es rep o no el número de bytes de la imatge
174     if numbytesaux1 != "":
175         print ("Enviem la correcta recepcio del nombre de bytes"+
176         de la imatge a rebre")
177         modem3.send('AT*SEND,2,2,OK')
178     acc = ""
179     #Llegim la imatge fins a trobar la exclamació que indica el final
180     #o el flag del "$" corresponent a la interrupció de la recepció
181     sortir = 0
182     while 1:
183               Imatge = modem3.port_read()
184               x1 = Imatge.find("!")
185               x2 = Imatge.find("$")
186               if x2!=-1:
187                   sortir=1
188                   print ("S'ha detingut la recepció d'imatges a" +
189                   "causa d'un error en l'enviament de les imatges"+
190                   "pero s'ha solucionat")
191                   break
192               if x1!=-1:
193                   acc = acc+Imatge.replace("!","")
194                   break
195               else:
196                   acc = acc+Imatge
197
198     #Es realitza un split per salts de línies per tal de separar els
199     #missatges i les notificacions en cas de no haver detectat
200     #l'error anterior
201     if sortir != 1:
202         Imatgeaux = acc.splitlines()
203         mida= len(Imatgeaux)
204         aux2 = 1
205         i = 1
206         ImatgeNova = ""
207         #Es separen les diferents línies amb comes per el posterior
208         #tractament
209         while i<mida:
210             if aux2 == 1:
211                 ImatgeNova = ImatgeNova + Imatgeaux[0]
212                 aux2 = 0
213             ImatgeNova = ImatgeNova + ',' + Imatgeaux[i]
214             i=i+1
215         #Es realitza un altre split per comes i s'elimina la
216         #informació dels "DELIVERED" o "FALSE" els quals no aporten
217         #informació
218         ImatgeNova2 = ImatgeNova.split(",")
219         #Es busquen tots els "RECV" que precedeixen a la informació
220         #del missatge i s'eliminen (la capçalera)
221         m = [u for u, x in enumerate(ImatgeNova2) if x == "RECV"]
222         i=0
223         ImatgeDef = ""
224         while i<len(m):
225             #S'agafen 9 posiciones més ja que el missatge es troba 9
226             #posicions més endevant del "RECV"
227             ImatgeDef = ImatgeDef + ImatgeNova2[m[i]+9]
228             i=i+1
```

```
229
230         #Es comprova si la imatge rebuda té el mateix número de bytes
231         #que l'original i es guarda en una carpeta amb la data actual
232         if len(ImatgeDef) == numbytes:
233             modem3.send('AT*SEND,2,2,OK')     #Es contesta amb un OK
234             print "La imatge ha sigut rebuda correctament"
235             dir = os.path.dirname(__file__)
236             save_path  = dir+'/'+str(datetime.date.today())
237             try:
238                 os.stat(save_path)
239             except:
240                 os.mkdir(save_path)
241             fitxer = os.path.join(save_path, str(tempsImatge)+".jpg")
242             fh=open(fitxer,"wb")
243             fh.write(ImatgeDef.decode('base64'))
244             fh.close
245
246         else:
247             modem3.send('AT*SEND,2,2,KO')
248             print "La imatge no ha sigut rebuda correctament"
249
250
```

A3. Programa versió 2: Enviament d'imatges automàtic

```
2   # -*- coding: utf-8 -*-
3   """
4   Created on Mon Feb 18 23:27:35 2019
5
6   @author: Oriol Solé
7
8   Programa versió2: Enviament d'imatges configurable (Programa modem
    transmissor)
9   """
10
11  #Llibreries#####################################################################
12
13  #Importació de les llibreries adients:
14  import base64   #Llibreria per codificar i descodificar les imatges en
15  #base 64
16  import socket   #Llibreria per fer us dels sockets
17  import time     #Llibreria per manipular tasques relacionades amb temps
18  import datetime #Llibreria per obtenir dates
19  import os       #Llibreria per manipular paths
20  import numpy    #Llibreria per executar operacions estadístiques
21  import xlsxwriter #Llibreria per operar amb fitxers Excel
22  from picamera import PiCamera #Llibreria per controlar la càmera de la
23  #Raspberry
24
```

```
25 #Definició Mòdems#####################################################
26
27 #Definició de les IPs i dels ports de comunicació dels mòdems acústics
28 #físics i virtuals transmissors
29 HOSTNAME1EM = '10.42.57.2'
30 HOSTNAME1 = '192.168.1.196'
31 PORT1 = 9200
32
33 #Classe funcions sockets##################################################
34
35 #Creació de la classe netcat per ubicar les funcions relatives a la
36 #manipulació dels sockets
37 class netcat(object):
38     def __init__ (self,hostname,port,modem_name):
39         #Temps d'espera per assegurar la connectivitat ethernet de la
40         #Raspberry al iniciar-se
41         time.sleep(10)
42         #Establiment de la ip, el port de comunicació i el nom del mòdem
43         self.ip = hostname
44         self.port = port
45         self.name = modem_name
46         #Creació del socket i connectivitat
47         try:
48             self.netcat = socket.socket(socket.AF_INET,
49             socket.SOCK_STREAM)
50             self.netcat.settimeout(0.1)
51             self.netcat.connect((self.ip, self.port))
52         except:
53             print "No s'ha pogut establir connexio"
54
55        #Configuracions inicials:
56         print 'Activacio permissos'
57         self.send('AT@CTRL')
58         print 'Desactivacio notificacions extesses'
59         self.send('AT@ZX0')
60         print 'Desactivacio notificacions de posicionament'
61         self.send('AT@ZU0')
62         print 'Reiniciament del buffer de transmissio:'
63         self.send('ATZ4')
64         print 'Configuracio del numero de bytes del buffer de'+
65         'transmissio:'
66         self.send('AT@ZL1000000')
67         print 'Guardar en memoria les configuracions del modem'
68         self.send('AT&W')
```

```
69
70      #Funcio de lectura de dades
71      def port_read(self):
72          buff = ''
73          aux = 0
74          read = ''
75          while 1:
76              #Bloc d'excepció, en cas de no poder establir connectivitat,
77              #i que no es tracti d'un error de timeout, es torna a intentar
78              #restablir la connexió perduda
79              try:
80                  read = self.netcat.recv(1)
81              except socket.error as error:
82                  if str(error)!= "timed out":
83                      try:
84                          self.netcat = socket.socket(socket.AF_INET,
85                          socket.SOCK_STREAM)
86                          self.netcat.settimeout(0.1)
87                          self.netcat.connect((self.ip, self.port))
88                          print "Connexio recuperada"
89                      except:
90                          print "Intent de connexio fallit"
91                          time.sleep(2)
92                  break
93              #Si es troba el flag de cancel·lació de l'enviament d'imatges
94              #"@" es retorna la variable aux per tal de interrompre
95              #l'enviament
96              x=read.find("@")
97              if x!=-1:
98                  aux=1
99              if read:
100                 buff += read
101             else:
102                 break
103         return buff,aux
104
105     #Funció per escriptura de comandes AT
106     def port_write(self,command):
107         while 1:
108             try:
109                 print 'Send',self.name +':',command
110                 self.netcat.sendall(command + '\n')
111                 break
112             except:
```

```
113                 try:
114                     self.netcat = socket.socket(socket.AF_INET,
115                     socket.SOCK_STREAM)
116                     self.netcat.settimeout(0.1)
117                     self.netcat.connect((self.ip, self.port))
118                     print "Connexio recuperada"
119                 except:
120                     print "Intent de connexio fallit"
121                     time.sleep(2)
122                     pass
123         return
124 
125     #Funció d'enviament de comandes AT amb comprovació de resposta
126     def send(self,command):
127         while 1:
128             self.port_write(command)
129             read = self.port_read()
130             x1 = read[0].find("BUSY CLOSING CONNECTION")
131             x2 = read[0].find("FAILED")
132             x3 = read[0].find("OK")
133             if x1!=-1:
134                 print "Received", self.name + ':', "FAILED"
135             elif x2!=-1:
136                 print "Received",self.name + ':', "BUSY CLOSING"+
137                 "CONNECTION"
138             elif x3!=-1:
139                 print "Received",self.name + ':', "OK"
140                 break
141             elif read[0] == "":
142                 print 'buit'
143                 time.sleep(2)
144             else:
145                 break
146             time.sleep(2)
147         return read[0],read[1]
148 
149     #Funció per el tancament del socket en cas d'interrupció del programa
150     def close(self):
151         self.netcat.shutdown(socket.SHUT_WR)
152         self.netcat.shutdown(socket.SHUT_RD)
153         self.netcat.close()
154 
155 #Funcions#################################################################
156 
```

```
157 #Funció per separar la imatge en un cert numero de bytes fora de la
158 #classe
159 def lista(foto,longitud):
160     return [foto[i:i + longitud] for i in range (0,len(foto), longitud)]
161
162
163 #Principal#################################################################
164
165 #Definició del mòdem amb la IP, el port de comunicació i el nom (crida
166 #de la classe netcat)
167 modem2 = netcat(HOSTNAME1,PORT1,'modem2')
168
169 #Obtenció del path on s'ubica el programa
170 dir = os.path.dirname(os.path.abspath(__file__))
171
172 #Definició variables inicials
173 llistatemps = []
174 llistatemps_aux = []
175 llistamissatge = []
176 llistafotos = []
177 NumeroSerie = 0
178 #Inicialització de la càmera
179 camera = PiCamera()
180 #Creacio del directori global
181 save_pathGlobal = dir+'/Results'
182 try:
183     os.stat(save_pathGlobal)
184 except:
185     os.mkdir(save_pathGlobal)
186 #Creació del directori amb les series on aniran les imatges
187 save_pathGlobalFotos = save_pathGlobal+'/Series'
188 try:
189     os.stat(save_pathGlobalFotos)
190 except:
191     os.mkdir(save_pathGlobalFotos)
192
193 #Bucle infinit Principal
194 while 1:
195     #Declaració i inicialització variables amb espera del numero de fotos
196     #a capturar, la resolució i la qualitat
197     NumeroSerie = NumeroSerie+1
198     aux4 = 0
199     img_correctes = 0
200     img=0
```

```
201     llistatemps=[]
202     llistamissatge=[]
203     llistafotos=[]
204     llistatemps_aux=[]
205     while 1:
206         info = modem2.port_read()
207         x= info[0].find("!")
208         if x!=-1:
209              info = info[0].replace("!","")
210              break
211
212     #Creació de les series amb les dates
213     save_pathSerien=save_pathGlobalFotos+'/'+
214     str((str(NumeroSerie)).zfill(4))+'_'+str(datetime.date.today())
215     #Es guarda la informació obtinguda en diferents variables
216     info = info.split(",")
217     Nfotos =int(info[9])
218     Resolucio1= int(info[10])
219     Resolucio2= int(info[11])
220     Qualitat = int(info[12])
221
222     #S'estableix la resolució i es crea bucle while per el numero de
223     #fotos a enviar
224     while img<Nfotos:
225         camera.resolution=(Resolucio1,Resolucio2)
226         try:
227             os.stat(save_pathSerien)
228         except:
229             os.mkdir(save_pathSerien)
230
231         #Es captura el temps per calcular la duració de l'enviament
232         temps=time.time()
233         color = 0
234         #Generació de fitxers d'errors i d'imatges
235         fichero = os.path.join(save_pathSerien,"MissatgeError.txt")
236         fichero1= os.path.join(save_pathSerien,str(temps)+".jpg")
237         #Captura de la imatge
238         camera.capture(save_pathSerien+'/'+str(temps)+'.jpg',
239         quality= Qualitat)
240         #Codificació de la imatge en base64
241         with open(save_pathSerien+'/'+str(temps)+'.jpg',"rb") as
242         imageFile:
243             nuevo = base64.b64encode(imageFile.read())
244
```

```
245         #Obtenció de la mida de la imatge, i creació de paquets de 1024
246         #bytes
247         valor = lista(nuevo,1024)
248         size = len(nuevo)
249 
250         #Enviament del numero de bytes de la imatge a transmetre
251         modem2.send('AT*SEND,'+str(len(str(size))+1)+',3,'+
252         str(size)+'#')
253         #Comprovació de la correcte recepció del numero de bytes
254         t = 0
255         while 1:
256             if aux4==1:
257                 t=50
258                 break
259             resposta2 = modem2.port_read()
260             x=resposta2[0].find("OK")
261             if resposta2[1]==1:
262                 aux4=1
263                 t=50
264                 break
265             if x!= -1:
266                 break
267             else:
268                 t = t+1
269                 if t == 100:
270                     break;
271 
272         #Si el mòdem respon, envia la imatge amb un flag final de
273         #exclamació
274         if t != 100:
275             i = 0
276         #S'envia la imatge i es comprova si l'usuari ha demanat la
277         #cancel·lació de les mateixes amb el flag "@"
278             while i<len(valor):
279                 if aux4 ==1:
280                     break
281                 resposta = modem2.port_read()
282                 if resposta[1] == 1:
283                     aux4=1
284                     break
285                 if i == len(valor)-1:
286                     resposta3 = modem2.send('AT*SEND,'+
287                     str(len(valor[i])+1)+',3,'+str(valor[i]+"!"))
288                     if resposta3[1]==1:
```

```
289                         aux4=1
290                         break
291                 else:
292                     resposta3 = modem2.send('AT*SEND,'+
293                     str(len(valor[i]))+',3,'+str(valor[i]))
294                     if resposta3[1]==1:
295                         aux4=1
296                         break
297                 i = i+1
298             t = 0
299             #Un cop es enviada la imatge s'espera la confirmació de la
300             #recepció OK" en cas d'haver sigut correctament enviada, "KO"
301             #en cas d'un enviament erroni si el mòdem receptor es
302             #desconnecta,"Disconnect" o si s'ha cancel·lat l'enviament
303             #d'imatges "Cancel"
304             while 1:
305                 if aux4 == 1:
306                     missatge = "Cancel"
307                     llistatemps_aux.append("")
308                     color = 1
309                     try:
310                         os.stat(save_pathSerien)
311                     except:
312                         os.mkdir(save_pathSerien)
313
314                     fh=open(fichero,"a")
315                     fh.write(str(temps)+': '+"La imatge no s'ha enviat"+
316                     "perque el client ha cancelat l'enviament de les"+
317                     "imatges"+'\n')
318                     fh.close
319                     fh =open(fichero1,"wb")
320                     fh.write(nuevo.decode('base64'))
321                     fh.close
322                     break
323                 #Si no es detecta la cancel·lació es comproven les
324                 #respostes
325                 resposta2 = modem2.port_read()
326                 x=resposta2[0].find("OK")
327                 x1=resposta2[0].find("KO")
328                 #Si la resposta es correcta, s'elimina la imatge
329                 if x!= -1:
330                     img_correctes = img_correctes+1
331                     missatge = "OK"
332                     os.system('rm '+save_pathSerien+'/'+
```

```
333                     str(temps)+'.jpg')
334                     tempsfinal = time.time()-temps
335                     llistatemps.append(tempsfinal)
336                     llistatemps_aux.append(tempsfinal)
337                     break
338                 else:
339                     #En cas d'incorrecte recepció, s'emmagatzema la
340                     #imatge, i es guarden els missatges d'error
341                     if x1!=-1:
342                         color=1
343                         missatge = "KO"
344                         llistatemps_aux.append("")
345                         try:
346                             os.stat(save_pathSerien)
347                         except:
348                             os.mkdir(save_pathSerien)
349 
350                         fh=open(fichero,"a")
351                         fh.write(str(temps)+': '+'El numero de'+
352                         'bytes de la imatge rebuda no coincideix amb'+
353                         'la original'+"\n")
354                         fh.close
355                         fh =open(fichero1,"wb")
356                         fh.write(nuevo.decode('base64'))
357                         fh.close
358                         break
359                      #Si no s'obté resposta, es comprova l'estat de
360                      #comunicació dels mòdems acústics i es finalitza
361                      #l'enviament, reiniciant el buffer de transmissió
362                     else:
363                         if resposta2[1]==1:
364                             aux4=1
365                         resposta = modem2.send("AT?S")
366                         aux1 =1
367                         print resposta
368                         x = resposta[0].find("INITIATION LISTEN")
369                         x2= resposta[0].find("1000000 bytes free")
370                         if x!=-1:
371                             aux1 =0
372                             modem2.send("ATZ4")
373                         if x2!=-1:
374                             t=t+1
375                         if resposta[1] == 1:
376                             aux4 = 1
```

```
377                         if t == 50 or aux1 == 0:
378                             missatge = "Disc."
379                             llistatemps_aux.append("")
380                             color = 1
381                             try:
382                                 os.stat(save_pathSerien)
383                             except:
384                                 os.mkdir(save_pathSerien)
385
386                             fh=open(fichero,"a")
387                             fh.write(str(temps)+': '+'El client es va'+
388                             'desconnectar mentres la imatge era'+
389                             'enviada'+'\n')
390                             fh.close
391                             fh =open(fichero1,"wb")
392                             fh.write(nuevo.decode('base64'))
393                             fh.close
394                             break;
395
396         #Si el client no contesta al cap d'un temps "t", es guarda la
397         #imatge i es genera un missatge d'error en el fitxer
398         else:
399             missatge = "OFF"
400             llistatemps_aux.append("")
401             color=1
402             try:
403                 os.stat(save_pathSerien)
404             except:
405                 os.mkdir(save_pathSerien)
406             fh=open(fichero,"a")
407             fh.write(str(temps)+': '+'El client estava desconnectat'+
408             '\n')
409             fh.close
410             fh =open(fichero1,"wb")
411             fh.write(nuevo.decode('base64'))
412             fh.close
413         img=img+1
414         llistamissatge.append(missatge)
415         llistafotos.append(temps)
416
417         if (aux4==1):
418             print 'Reset Buffer transmission:'
419             modem2.send('ATZ4')
420             break
```

```
421
422     #En cas d'haver completat el numero de fotos previstes d'enviar:
423     if len(llistatemps) == 0:
424         temps_mitja = ""
425         desviacio_estandard = ""
426         percentatge= "0"
427     else:
428
429         temps_mitja = sum(llistatemps)/len(llistatemps)
430         desviacio_estandard = numpy.std(llistatemps)
431         percentatge = (img_correctes*100)/img
432
433     try:
434         os.stat(save_pathSerien)
435     except:
436         os.mkdir(save_pathSerien)
437
438     #Creació de arxius Excel amb informació relativa al enviament de
439     #imatges
440     workbook=xlsxwriter.Workbook(save_pathSerien+
441     '/fitxerEstadistic.xlsx')
442     worksheet = workbook.add_worksheet()
443     row = 1
444     i=0
445     bold = workbook.add_format({'bold': True})
446     red = workbook.add_format({'color':'red'})
447
448     worksheet.write(0,0,'Nom Imatge',bold)
449     worksheet.write(0,1,'Temps en (s)',bold)
450     worksheet.write(0,2,'Estat Imatge',bold)
451     worksheet.write(0,3,'Temps mitja en (s)',bold)
452     worksheet.write(1,3,temps_mitja)
453     worksheet.write(0,4,'Desviacio estandard',bold)
454     worksheet.write(1,4,desviacio_estandard)
455     worksheet.write(0,5,'Percentaje de fotos enviades',bold)
456     worksheet.write(1,5,percentatge)
457     worksheet.write(0,6,'Numero de mostres',bold)
458     worksheet.write(1,6,img)
459
460     row = 1
461
462     for temps in (llistatemps_aux):
```

```
463         worksheet.write(row,1,temps)
464         row+=1
465 
466     row = 1
467 
468     for missatges in (llistamissatge):
469         x1 = missatges.find("OK")
470         if x1!=-1:
471             worksheet.write(row,2,missatges)
472             worksheet.write(row,0,llistafotos[i])
473         else:
474             worksheet.write(row,2,missatges,red)
475             worksheet.write(row,0,llistafotos[i],red)
476         row+=1
477         i+=1
478     workbook.close()
479 
480     #S'envia el flag per avisar al modem receptor de que s'ha finalitzat
481     #amb l'enviament de les imatges previstes
482     modem2.send('AT*SEND,1,3,$')
483 modem2.close()
```

A4. Programa versió 2: Recepció d'imatges automàtic

```
2   # -*- coding: utf-8 -*-
3   """
4   Created on Sat Feb 16 13:40:41 2019
5 
6   @author: Oriol Solé
7 
8   Programa versió2: Enviament d'imatges configurable (Programa modem
    receptor)
9   """
10 
484 #Llibreries#################################################################
485 
486 #Importació de les llibreries adients:
11  import base64   #Llibreria per codificar i descodificar les imatges
12  #en base 64
13  import socket   #Llibreria per fer ús dels sockets
14  import time     #Llibreria per manipular tasques relacionades amb
15   #temps
16  import datetime #Llibreria per obentir dates
17  import os       #Llibreria per manipular paths
18  import sys      #Llibreria per executar instruccions amb permis del
19  #sistema
20 
21  #Configuració Mòdems#############################################
22  #Definició de les IPs i dels ports de comunicació dels mòdems
```

```
23 #acústics físics i virtuals receptors
24 HOSTNAME2EM = '10.42.57.3'
25 HOSTNAME2 = '192.168.1.197'
26 PORT2 = 9200
27
28 #Classe funcions sockets##############################################
29 condicioaux = 0
30 #Creació de la classe netcat per ubicar les funcions relatives a la
31 #manipulació dels sockets
32
33 class netcat(object):
34     #Definició de la funció inicial per la configuració del mòdem
35     #acústic
36     def __init__ (self,hostname,port,modem_name):
37         #Establiment de la ip, el port de comunicació i el nom del
38         #mòdem
39         self.ip = hostname
40         self.port = port
41         self.name = modem_name
42         #Creació del socket i connectivitat
43         try:
44             self.netcat = socket.socket(socket.AF_INET,
45             socket.SOCK_STREAM)
46             self.netcat.settimeout(0.1)
47             self.netcat.connect((self.ip, self.port))
48         except:
49             print ("El programa es tancara, ja que no s'ha pogut"+
50             "establir una connexió degut a que el sistema no esta"+
51             "alimentat, connecti la alimentacio i executi de nou el"+
52             "programa")
53             time.sleep(5)
54             sys.exit(1)
55
56         #Comprovar si s'ha rebut el flag "$" mentres el programa
57         #estava tancat abans de reiniciar el buffer de transmissió
58         global condicioaux
59
60         resposta_aux = self.port_read()
61         if resposta_aux[1] == 1:
62             condicioaux = 1
63
64         #Configuracions inicials:
65         print 'Activacio permissos'
66         self.send('AT@CTRL')
67         print 'Desactivacio notificacions extesses'
68         self.send('AT@ZX0')
69         print 'Desactivacio notificacions de posicionament'
70         self.send('AT@ZU0')
71         print 'Reiniciament del buffer de transmissio:'
72         self.send('ATZ4')
73         print 'Configuracio del numero de bytes del buffer de'+
74         'transmissio:'
75         self.send('AT@ZL1000000')
76         print 'Guardar en memoria les configuracions del modem'
77         self.send('AT&W')
78
79     #Funció de lectura de dades
80     def port_read(self):
```

```
81          buff = ''
82          aux = 0
83          read = ''
84          while 1:
85              #Bloc d'excepció, en cas de no poder establir
86              #connectivitat, i que no es tracti d'un error de timeout,
87              #es torna a intentar restablir la connexió perduda
88              try:
89                  read = self.netcat.recv(1)
90              except socket.error as error:
91                  if str(error) != "timed out":
92                      print ("El programa es tancara degut a un"+
93                      "problema amb la alimentacio, comprovala i"+
94                      "executa de nou el programa")
95                      time.sleep(5)
96                      sys.exit(1)
97                      break
98                  break
99              if read:
100               buff += read
101               #S'observa si mentres s'ha llegit el buffer del mòdem,
102               #paral·lela-ment s'ha rebut el flag "$"
103               valor = read.find("$")
104               if valor!=-1:
105                 aux = 1
106             else:
107                 break
108         return buff,aux
109
110     #Funció per escriptura de comandes AT
111     def port_write(self,command):
112        while 1:
113            print 'Send',self.name +':',command
114            try:
115                self.netcat.sendall(command + '\n')
116                break
117            except:
118                try:
119                    self.netcat = socket.socket(socket.AF_INET,
120                    socket.SOCK_STREAM)
121                    self.netcat.settimeout(0.1)
122                    self.netcat.connect((self.ip, self.port))
123                    print "Connexio recuperada"
124                except:
125                    print "Intent de connexio fallit"
126                    time.sleep(2)
127                    pass
128        return
129
130     #Funció d'enviament de comandes AT amb comprovació de resposta
131     def send(self,command):
132         while 1:
133             try:
134                 self.port_write(command)
135                 read = self.port_read()
136                 x1 = read[0].find("BUSY CLOSING CONNECTION")
137                 x2 = read[0].find("FAILED")
138                 x3 = read[0].find("OK")
```

```
139                 if x1!=-1:
140                     print "Received", self.name + ':', "FAILED"
141                 elif x2!=-1:
142                     print "Received",self.name + ':', "BUSY CLOSING"+
143                     "CONNECTION"
144                 elif x3!=-1:
145                     print "Received",self.name + ':', "OK"
146                     break
147                 elif read[0] == "":
148                     print 'buit'
149                     time.sleep(2)
150                 else:
151                     break
152             except:
153                 break;
154             time.sleep(2)
155         return read[0],read[1]
156 
157     #Funció per el tancament del socket en cas d'interrupció del
158     #programa
159     def close(self):
160         self.netcat.shutdown(socket.SHUT_RD)
161         self.netcat.close()
162 
163 
164 #Funcions#################################################################
165 
166 #Funció basada en el programa "Read_PC", la qual se li ha de passar
167 #com a parametres, el directori en el qual ha de guardar la imatge
168 
169 def Read_Foto(save_pathSerien):
170     numbytesaux = ""
171     aux = 0
172     timer = 0
173     #Bucle per l'espera del numero de bytes a rebre amb el flag "#"
174     while 1:
175         numbytesaux = modem3.port_read()
176         #Si es troba el flag "$" es retorna el valor auxiliar dat,
177         #per indicar la recepció de la finalització de la captura i
178         #enviament de les imatges per part del mòdem transmissor
179         z = numbytesaux[0].find("#")
180         if z!= -1 and numbytesaux[1] == 1:
181             aux=1
182             return aux
183         elif z!=-1:
184             numbytesaux = numbytesaux[0].replace("#","")
185             tempsImatge = time.time()
186             break
187         elif numbytesaux[1] == 1:
188             aux = 1
189             return aux
190     numbytesaux1 = numbytesaux.split(",")
191     numbytes = int(numbytesaux1[-1])
192     print "El numero de bytes de la imatge son: "+str(numbytes)
193     time.sleep(2)
194 
195     #Es contesta si es rep o no el número de bytes de la imatge
196     if numbytesaux1 != "":
```

```
197         print ("Enviem la correcta recepcio del nombre de bytes de"+
198         "la imatge a rebre")
199         modem3.send('AT*SEND,2,2,OK')
200 
201     acc = ""
202     sortir = 0
203     #Llegim la imatge fins a trobar la exclamació que indica el final
204     #o el flag del "$" corresponent a la interrupció
205     while 1:
206           Imatge = modem3.port_read()
207           x1 = Imatge[0].find("!")
208           if Imatge[1] == 1:
209               sortir=1
210               return sortir
211           if x1!=-1:
212               acc = acc+Imatge[0].replace("!","")
213               break
214           else:
215               acc = acc+Imatge[0]
216 
217     #Es realitza un split per salts de línies per tal de separar
218     #missatges i les notificacions
219     Imatgeaux = acc.splitlines()
220     mida = len(Imatgeaux)
221     aux2 = 1
222     i = 1
223     ImatgeNova = ""
224     #Es separen les diferents línies amb comes per el posterior
225     #tractament
226     while i<mida:
227         if aux2 == 1:
228             ImatgeNova = ImatgeNova + Imatgeaux[0]
229             aux2 = 0
230         ImatgeNova = ImatgeNova + ',' + Imatgeaux[i]
231         i=i+1
232 
233     #Es realitza un altre split per comes i s'elimina la informació
234     #dels "DELIVERED" o "FALSE" els quals no aporten informació
235     ImatgeNova2 = ImatgeNova.split(",")
236 
237     #Es busquen tots els "RECV" que precedeixen a la informació del
238     #missatge i s'eliminen (la capçalera)
239     m = [u for u, x in enumerate(ImatgeNova2) if x == "RECV"]
240     i=0
241     ImatgeDef = ""
242     while i<len(m):
243         #S'agafen 9 posiciones més ja que el missatge es troba 9
244         #posicions més endevant del "RECV"
245         ImatgeDef = ImatgeDef + ImatgeNova2[m[i]+9]
246         i=i+1
247 
248     #Es comprova si la imatge rebuda té el mateix número de bytes que
249     #l'original i es guarda en una carpeta amb la data actual
250     if len(ImatgeDef) == numbytes:
251         modem3.send('AT*SEND,2,2,OK')     #Es contesta amb un OK
252         print "La imatge ha sigut rebuda correctament"
253         time.sleep(5)
254 
```

```
255         try:
256             os.stat(save_pathSerien)
257         except:
258             os.mkdir(save_pathSerien)
259 
260         fichero = os.path.join(save_pathSerien, str(tempsImatge)+
261         ".jpg")
262         fh=open(fichero,"wb")
263         fh.write(ImatgeDef.decode('base64'))
264         fh.close
265 
266     else:
267         modem3.send('AT*SEND,2,2,KO')
268         print "La imatge no ha sigut rebuda correctament"
269         time.sleep(5)
270 
271 #Principal#################################################################
272 
273 #Definició del mòdem amb la IP, el port de comunicació i el nom
274 #(crida de la classe netcat)
275 
276 modem3 = netcat(HOSTNAME2,PORT2,'modem3')
277 
278 #Obtenció del path on s'ubica el programa i creació de directoris per
279 #ubicar les diferents peticions de series d'imatges que l'usuari
280 #demana
281 
282 dir = os.path.dirname(__file__)
283 save_pathGlobal = dir+'/Results'
284 
285 try:
286     os.stat(save_pathGlobal)
287 except:
288     os.mkdir(save_pathGlobal)
289 
290 save_pathGlobalFotos = save_pathGlobal+'/Series'
291 
292 try:
293     os.stat(save_pathGlobalFotos)
294 except:
295     os.mkdir(save_pathGlobalFotos)
296 
297 #Bucle infinit principal
298 while 1:
299     #Creació d'un fitxer de text per tal de guardar el número de
300     #serie pel qual es va, per disposar-lo en memòria en cas de
301     #tancament del programa
302     fitxerserie = os.path.join(save_pathGlobal,"NumeroSerie.txt")
303     respostax = os.path.exists(fitxerserie)
304     #En cas de no existir, es crea i es comença amb la primera serie
305     if respostax == False:
306         fh = open(fitxerserie,"w")
307         fh.write("1")
308         fh.close()
309         nserie = 1
310     #Si existeix, s'obre es llegeix i es continua amb la serie
311     #indicada
312     else:
```

```
313         fh = open(fitxerserie,"r")
314         nseriex = fh.read()
315         fh.close()
316         nserie = int(nseriex)
317 
318     #Si existien imatges que havien sigut demanades anteriorment, i a
319     #causa del tancament del programa no es van poder rebre totes, es
320     #dona l'opció de continuar amb la recepció de les mateixes, o
321     #cancelar-les. Sinò, es comença una nova serie
322 
323     fitxerfotos = os.path.join(save_pathGlobal,"FotosPrevistes.txt")
324     resposta = os.path.exists(fitxerfotos)
325     #Començament serie nova: Nfotos a capturar, resolució i qualitat
326 
327     if resposta == False:
328         while 1:
329             fotos = raw_input("Introdueix el numero de fotos a"+
330             "capturar: ")
331             numero = fotos.isdigit()
332             if numero == True:
333                 Nfotos = int(fotos)
334                 if Nfotos>0:
335                     break
336                 else:
337                     print "El propòsit del programa és capturar"+
338                     "imatges..."
339             else:
340                 print "El valor no és vàlid"
341 
342         while 1:
343             resolucioaux = raw_input("Introdueix la resolució: ")
344             x1=resolucioaux.find("640x480")
345             x2=resolucioaux.find("720x480")
346             x3=resolucioaux.find("1280x720")
347             if x1!=-1 or x2!=-1 or x3!=-1:
348                 break
349             else:
350                 print "El valor no és vàlid"
351         resolucio = resolucioaux.split("x")
352 
353         while 1:
354             qualitat = raw_input("Introdueix la qualitat d'imatge: ")
355             numero = qualitat.isdigit()
356             qualitat = int(qualitat)
357             if numero == True and qualitat>0 and qualitat<=100:
358                 break
359             else:
360                 print "El valor no és vàlid"
361 
362         #Es guarda el número de fotos previstes a capturar, per si es
363         #produeix un tancament del programa
364         fh = open(fitxerfotos,"w")
365         fh.write(fotos)
366         fh.close()
367 
368         info = ""
369         info=str(Nfotos)+","+resolucio[0]+","+resolucio[1]+","+
370         str(qualitat)
```

```
371         time.sleep(2)
372
373         #S'envia la informació del numero de fotos, resolució i
374         #qualitat
375         modem3.send('AT*SEND,'+str(len(str(info))+1)+',2,'+info+'!')
376         #S'executa la funció de llegir tantes vegades com nfotos hi
377         #hagi
378
379         #Si el mòdem transmissor falla, es detecta a partir del flag
380         #"$"
381         NumeroFotos = 0
382         save_pathSerien = save_pathGlobalFotos+'/'+
383         str((str(nserie)).zfill(4))+'_'+str(datetime.date.today())
384         while NumeroFotos<Nfotos:
385             sortir = Read_Foto(save_pathSerien)
386             if sortir==1:
387                 print ("Hi ha hagut un error en el modem"+
388                 "transmissor pero s'ha restablert")
389                 break
390             NumeroFotos=NumeroFotos+1
391         while 1:
392             valor = modem3.port_read()
393             if valor[1] == 1 or sortir == 1 or sortir == 2:
394                 os.remove(save_pathGlobal+'/FotosPrevistes.txt')
395                 nserie = nserie+1
396                 fh = open(fitxerserie,"w")
397                 fh.write(str(nserie))
398                 fh.close()
399                 break
400
401     #Si existien fotos previstes, primer es comprova si el mòdem ja
402     #les havia enviat, i sino es permet la opció de seguir rebent-les
403     #o cancelar-les
404
405     elif resposta == True:
406         valor = modem3.port_read()
407         if valor[1] == 1:
408             os.remove(save_pathGlobal+'/FotosPrevistes.txt')
409             nserie = nserie+1
410             fh = open(fitxerserie,"w")
411             fh.write(str(nserie))
412             fh.close()
413
414         else:
415
416             fh = open(fitxerfotos,"r")
417             nfotos = fh.read()
418             fh.close()
419             r = 0
420             z=0
421             condicio2 = 0
422             while 1:
423                 Delete = raw_input("Tens "+nfotos+" fotos previstes"+
424                 "de capturar, vols cancerlar-les (1) o seguir igual"+
425                 "(0): ")
426                 #Si es volen seguir rebent:
427                 if Delete == "0":
428                     condicio = 0
```

```
429                     print ("Es procedeix a seguir rebent les imatges"
430                     +" comprovant l'estat del modem")
431                     while 1:
432                         #Es comprova l'estat del mòdem, per veure si
433                         #l'altre mòdem ja ha finalitzat, ha enviat el
434                         #flag, pero el mòdem receptor estava
435                         #desconnectat, i per tant, no ha pogut rebre
436                         #el flag
437                         while 1:
438                             respostax = modem3.send("AT?S")
439                             #Si el modem no es troba online,
440                             #significa que el mòdem transmissor ja ha
441                             #acabat l'enviament de les imatges
442                             x1 = respostax[0].find("INITIATION" +
443                             "LISTEN")
444                             if respostax[1] == 1:
445                                 condicio2 = 1
446                                 break
447 
448                             if x1!=-1:
449                                 z=z+1
450                                 time.sleep(2)
451                                 if z==10:
452                                     condicio2 = 1
453                                     break
454                             else:
455                                 break
456 
457                         #s'eliminen les fotos previstes, en cas
458                         #d'haver acabat l'enviament el mòdem
459                         #transmissor
460                         valor = modem3.port_read()
461                         if (valor[1]== 1 or condicio == 1 or
462                         condicio2 == 1 or condicioaux == 1):
463                             os.remove(save_pathGlobal+
464                             '/FotosPrevistes.txt')
465                             print("Les imatges ja han estat"+
466                             "capturades i processades")
467                             break
468                         #sino es torna a llegir el número de bytes de
469                         #la imatge següent
470                         save_pathSerien = save_pathGlobalFotos+'/'\
471                         +str((str(nserie)).zfill(4))+'_'\
472                         +str(datetime.date.today())
473                         condicio = Read_Foto(save_pathSerien)
474                     nserie = nserie+1
475                     fh = open(fitxerserie,"w")
476                     fh.write(str(nserie))
477                     fh.close()
478                     break
479                 #Si es volen eliminar les imatges, primer es comprova
480                 #com en l'altre cas, si el mòdem ja ha finalitzat
481                 #l'enviament
482                 elif Delete == "1":
483                     while 1:
484                         i=0
485                         aux2 = 0
486                         while i<10:
```

```
487                             valor= modem3.port_read()
488                             if valor[1]== 1 or condicioaux == 1:
489                                 os.remove(save_pathGlobal+
490                                 '/FotosPrevistes.txt')
491                                 print("La captura de imatges ha"+
492                                 "sigut detinguda")
493                                 aux2=1
494                                 break
495                             i=i+1
496                         aux = 1
497                         if aux2== 1:
498                             break
499 
500                         #Per tal d'assegurar la finalització de la
501                         #captura es comprova tant l'estat del mòdem,
502                         #com s'envia un flag "@" per tal de comunicar
503                         #al mòdem la finalització de la serie
504 
505                         while 1:
506                             aux4=0
507                             respostax = modem3.send("AT?S")
508                             x1 = respostax[0].find("INITIATION"+
509                             "LISTEN")
510                             if x1!=-1:
511                                 aux4=1
512                             if respostax[1]== 1:
513                                 aux4 = 1
514                             print ("Es procedeix a comunicar al"+
515                             "modem transmissor la cancelacio del"+
516                             "enviament de les imatges")
517                             resposta = modem3.send('AT*SEND,1,2,@')
518                             time.sleep(5)
519                             if resposta[1] == 1:
520                                 aux4=1
521 
522                             valor2 = modem3.port_read()
523                             if valor2[1] == 1 or aux4==1 or
524                             condicioaux==1:
525                                 os.remove(save_pathGlobal+
526                                 '/FotosPrevistes.txt')
527                                 time.sleep(10)
528                                 print("La captura de imatges ha"+
529                                 "sigut detinguda")
530                                 break
531                         break
532                     nserie = nserie+1
533                     fh = open(fitxerserie,"w")
534                     fh.write(str(nserie))
535                     fh.close()
536                     break
537                 else:
538                     print("El valor introducido no és vàlid,torna a"+
539                     "introduir-lo: ")
540 
541 modem3.close()
```